Oracle® Clusterware

Administration and Deployment Guide

11*g* Release 1 (11.1)

B28255-01

July 2007

ORACLE®

Oracle Clusterware Administration and Deployment Guide, 11*g* Release 1 (11.1)

B28255-01

Primary Author: Viv Schupmann

Contributing Authors: Ahmed Abbas, Mark Bauer, Eric Belden, Jonathan Creighton, Andrey Gusev, Winston Huang, Rajiv Jayaraman, Sameer Joshi, Roland Knapp, Erich Kreisler, Raj Kumar, Ivan Lam, Karen Li, Barb Lundhild, Bill Manry, Markus Michalewicz, Anil Nair, Philip Newlan, Kevin Reardon, Saar Maoz, K.P. Singh, Duane Smith, Alok Srivastava, Janet Stern, Richard Strohm, Su Tang, Juan Tellez, Ramkumar Venkatesan, Douglas Williiams

Contents

A Cluster Verification Utility Reference

B OLSNODES Command Reference

C Oracle Interface Configuration (OIFCFG) Command Reference

D High Availability Oracle Clusterware Command-Line Reference and C API

Preface

The *Oracle Clusterware Administration and Deployment Guide* describes the Oracle Clusterware architecture and provides an overview of this product. This book also describes administrative and deployment topics for Oracle Clusterware.

Information in this manual applies to Oracle Clusterware as it runs on all platforms unless otherwise noted. In addition, the content of this manual supplements administrative and deployment topics for Oracle single-instance databases that appear in other Oracle documentation. Where necessary, this manual refers to platform-specific documentation. This Preface contains these topics:

Audience

The *Oracle Clusterware Administration and Deployment Guide* is intended for database administrators, network administrators, and system administrators who perform the following tasks:

- Install and configure Oracle RAC databases
- Administer and manage Oracle RAC databases
- Manage and troubleshoot clusters and networks that use Oracle RAC

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the Oracle resources listed in this section.

- Platform-specific Oracle Clusterware and Oracle RAC installation guides

  Each platform-specific Oracle Database 11*g* installation media contains a copy of an Oracle Clusterware and Oracle RAC platform-specific installation and configuration guide in HTML and PDF formats. These installation books contain the preinstallation, installation, and post-installation information for the various UNIX, Linux, and Windows platforms on which Oracle Clusterware and Oracle RAC operate.

- *Oracle Database 2 Day + Real Application Clusters Guide*

  This task-oriented guide helps you understand the basic steps required to install, configure, and administer Oracle Clusterware and Oracle Real Application Clusters on a two-node system using Red Hat Linux system.

- *Oracle Real Application Clusters Administration and Deployment Guide*

  This is an essential companion book that describes Oracle Clusterware components such as the voting disks and the Oracle Cluster Registry (OCR).

- *Oracle Database 2 Day DBA*
- *Oracle Database Administrator's Guide*
- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Platform Guide for Microsoft Windows*
- *Oracle Database Administrator's Reference 11g Release 1 (11.1) for UNIX Operating Systems: AIX Systems, HP-UX, Linux, and the Solaris Operating System (SPARC)*

Database error messages descriptions are available online or by way of a Tahiti documentation search.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in Oracle Clusterware Administration and Deployment?

This section describes the new administration and deployment features for Oracle Clusterware in Oracle Database 11*g* Release 1 (11.1).

Oracle Database 11g Release 1 (11.1) New Features in Oracle Clusterware

This section describes the Oracle Database 11*g* release 1 (11.1) features for Oracle Clusterware administration and deployment.

- **Oracle Clusterware Documented Separately**

  Oracle Clusterware is now documented in its own book. This book, *Oracle Clusterware Administration and Deployment Guide*, describes how to administer and deploy Oracle Clusterware. This book also contains information about the Oracle Clusterware API and API commands.

  See Also: Your platform-specific Oracle Clusterware installation guide for more information about installing Oracle Clusterware

- **Oracle cloning procedures for Oracle Clusterware**

  This book includes new step-by-step procedures for cloning Oracle Clusterware homes to quickly create Oracle Clusterware environments on other nodes.

  See Also: Chapter 3, "Cloning Oracle Clusterware"

- **Optimal Flexible Architecture (OFA) ensures reliable installations and improves software manageability**

  This feature improves manageability by making default Oracle Database installations more compliant with Optimal Flexible Architecture (OFA) specifications. As a part of this feature, the Oracle Universal Installer has been updated so that the default installation follows Oracle's Optimal Flexible Architecture. This ensures reliable installations and improves software manageability.

  In addition, Oracle Clusterware can be installed only once on the system, so a counter to track the number of times that Oracle Clusterware is installed is not required as it is for Oracle Database homes and Oracle installations with Oracle RAC.

See Also: "About the Oracle Clusterware Installation" and your platform-specific Oracle Clusterware and Oracle Real Application Clusters installation and configuration guide

- **New `ocrconfig` command options**

  The `ocrconfig` command now includes a new `-manualbackup` option that enables you to force a manual backup at any time, rather than wait for the automatic backup that occurs at 4 (or more) hour intervals. The new `-manualbackup` option provides a method for obtaining a binary backup on demand, such as after you make changes to the Oracle Cluster Registry (OCR).

  In addition, the existing `-showbackup` option includes new `auto` and `manual` flags that you can optionally specify to display only the automatic backup information or manual backup information, respectively.

  See Also: "Managing Backups and Recovering the Oracle Cluster Registry" on page 2-7 and Appendix E, "Oracle Cluster Registry Configuration Tool Command Reference"

- **Add voting disks dynamically, with no downtime**

  You can now add a voting disk while the cluster is active. Cluster Synchronization Services (CSS) start to use the voting disk without the need to restart the cluster.

- **Improved Oracle Clusterware performance monitoring and diagnostics in Enterprise Manager**

  Both Oracle Enterprise Manager Database Control and Oracle Enterprise Manager Grid Control are cluster aware and provide a central console to manage your cluster database. From any location where you can access a web browser, you can manage Oracle Clusterware, application servers, host computers, and Web applications, as well as related hardware and software.

  You now have the ability to see any given metric across database instances or hosts in the cluster as a tile chart. This high-level view capability means that you can review issues that are affecting the entire cluster as well as those that are affecting individual instances. This is a major enhancement in terms of how metrics are monitored for Oracle Clusterware. With Oracle 11*g*, you can see the roll-up or summary-based views as well as tile based views if you want to monitor how a metric performs across different hosts over a period of time.

  See Also: Monitoring Oracle Clusterware on page F-1 and *Oracle Database 2 Day + Real Application Clusters Guide*

1

Introduction to Oracle Clusterware

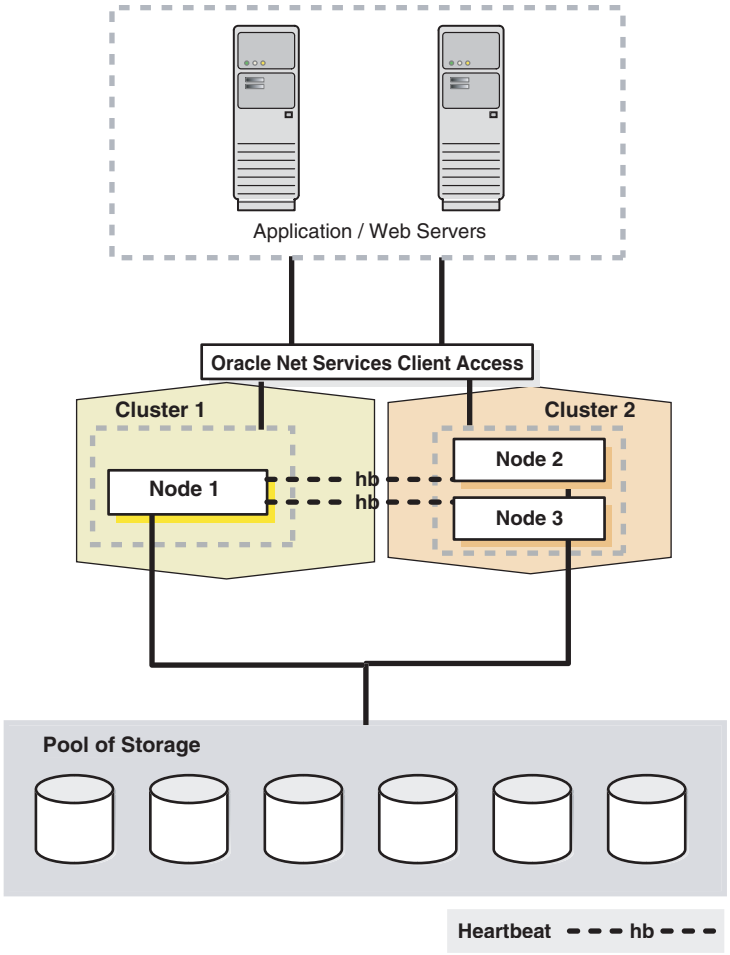
This chapter introduces Oracle Clusterware and describes how to install, administer, and deploy it. This chapter includes the following topics:

What is Oracle Clusterware?

Oracle Clusterware is software that enables servers to operate together as if they are one server. Each server looks like any standalone server. However, each server has additional processes that communicate with each other so the separate servers appear as if they are one server to applications and end users.

Figure 1–1 shows a configuration that uses Oracle Clusterware to extend the basic single-instance Oracle Database architecture. In the figure, both Cluster 1 and Cluster 2 are connected to the Oracle Database and are actively servicing applications and users. Using Oracle Clusterware, you can use the same high availability mechanisms to make your Oracle database and your custom applications highly available.

Figure 1–1 Oracle Clusterware Configuration



The benefits of using a cluster include:

- Scalability for applications
- Using lower-cost hardware
- Ability to fail over
- Ability to grow the capacity over time by adding servers, when needed

You can program Oracle Clusterware to manage the availability of user applications and Oracle databases.

In an Oracle Real Application Clusters (Oracle RAC) environment, Oracle Clusterware manages all of the Oracle processes automatically. Anything that Oracle Clusterware manages is known as a cluster resource, which could be a database, an instance, a service, a listener, a virtual IP (VIP) address, an application process, and so on.

Creating a cluster with Oracle Clusterware provides the ability to:

- Eliminate unplanned downtime due to a hardware or software malfunction
- Reduce or eliminate planned downtime for software maintenance
- Increase throughput for applications by allowing the applications to run on all of the nodes in the cluster

- Increase throughput for the application, as required, by adding servers to the cluster, when necessary
- Reduce the total cost of ownership for infrastructure by providing a scalable system with low cost commodity hardware

Oracle Clusterware is a requirement for using Oracle RAC and it is the only clusterware that you need for most platforms on which Oracle RAC operates. Although Oracle RAC continues to support select third-party clusterware products on specific platforms, you must also install and use Oracle Clusterware. Note that the servers on which you want to install and run Oracle Clusterware must be running the same operating system.

Using Oracle Clusterware eliminates the need for proprietary vendor clusterware and provides the benefit of using only Oracle software. Oracle provides an entire software solution, including everything from disk management with Oracle Automatic Storage Management (ASM) to data management with the Oracle Database and Oracle RAC. In addition, Oracle Database features, such as Oracle Services, provide advanced features when used with the underlying Oracle Clusterware high availability framework.

Oracle Clusterware requires two components: a voting disk to record node membership information and the Oracle Cluster Registry (OCR) to record cluster configuration information. The voting disk and the OCR must reside on shared storage.

To use and install Oracle Clusterware, you need to understand the hardware and software concepts and requirements, as described in the following sections:

- Oracle Clusterware Hardware Concepts and Requirements
- Oracle Clusterware Software Concepts and Requirements

Oracle Clusterware Hardware Concepts and Requirements

Many hardware providers have validated cluster configurations that provide a single part number for a cluster. If you are new to clustering, the information in this section will make the hardware procurement easier when you work with hardware vendors to purchase the appropriate hardware to create a cluster.

A cluster is made up of one or more servers. A server in a cluster is similar to any standalone server, but a cluster requires a second network called the interconnect network. Therefore, the server minimally requires two network interface cards: one for the public network and one for the private network. The interconnect network is a private network using a switch (or multiple switches) that only the nodes in the cluster can access.[1] Crossover cables are not supported for use with Oracle Clusterware interconnects.

The size of the server is dictated by the requirements of the workload you want to run on the cluster and the number of nodes you have chosen to configure in the cluster. If you are implementing the cluster for high availability, then configure redundancy for all components of the infrastructure. Therefore, you need to configure:

- A network interface for the public network (generally this is an internal LAN)
- A redundant network interface for the public network
- A network interface for the private interconnect network

[1] Oracle Clusterware supports up to 100 nodes in a cluster on configurations running Oracle Database 10*g* Release 2 and later releases.

- A redundant network interface for the private interconnect network

The cluster requires cluster-aware storage[2] that is connected to each server in the cluster. This may also be referred to as a multihost device. Oracle supports both Storage Area Network (SAN) storage or Network Attached (NAS) storage.

Similar to the network, there are generally at least two connections from each server to the cluster-aware storage to provide redundancy. There may be more connections depending on your I/O requirements. It is important to consider the I/O requirements of the entire cluster when choosing the storage subsystem.

Most servers have at least one local disk that is internal to the server. Often, this disk is used for the operating system binaries and you can also use this disk for the Oracle binaries. The benefit of each server having its own copy of the binaries is that it simplifies rolling upgrades.

Oracle Clusterware Software Concepts and Requirements

Oracle Clusterware uses a shared common disk for its configuration files.

Oracle Clusterware requires two configuration files: a voting disk to record node membership information and the OCR to record cluster configuration information. During the Oracle Clusterware installation, Oracle recommends that you configure multiple voting disks and the OCR:

- **Voting Disk**

  Oracle Clusterware uses the voting disk to determine which instances are members of a cluster. The voting disk must reside on a shared disk. For high availability, Oracle recommends that you have a minimum of three voting disks. If you configure a single voting disk, then you should use external mirroring to provide redundancy. You can have up to 32 voting disks in your cluster.

- **Oracle Cluster Registry (OCR)**

  Oracle Clusterware uses the OCR to store and manage information about the components that Oracle Clusterware controls, such as Oracle RAC databases, listeners, VIPs, and services and any applications. The OCR repository stores configuration information in a series of key-value pairs in a directory tree structure.

  Oracle recommends that you use a multiplexed OCR to ensure cluster high availability. Consider the following points regarding the OCR:

  – The OCR must reside on a shared disk that is accessible by all of the nodes in the cluster.

  – Oracle Clusterware can multiplex the OCR.

  – You can replace a failed OCR online.

  – You must update the OCR through supported APIs such as Enterprise Manager, the Server Control Utility (SRVCTL), or the Database Configuration Assistant (DBCA).

  – Oracle Clusterware requires that each node be connected to a private network by way of a private interconnect. For redundancy, you can have up to 32 voting disks and a mirror of the OCR.

[2] Cluster-aware storage may also be referred to as a multihost device.

See Also: Chapter 2, "Administering Oracle Clusterware" for more information about voting disks and the OCR

Oracle Clusterware requires a virtual IP address for each node in the cluster. This IP address must be on the same subnet as the public IP address for the node and should be an address that is assigned a name in the Domain Name Service, but is unused and cannot be *pinged* in the network before installation of Oracle Clusterware. The VIP is a node application (nodeapp) defined in the OCR that is managed by Oracle Clusterware. The VIP is configured with the VIPCA utility. The root script calls the VIPCA utility in silent mode.

Each server must first have an operating system that is certified with the Oracle Clusterware version you are installing. See the certification matrices available on Oracle Metalink (`http://certify.oraclecorp.com/certifyv3/certify/cert_views.group_selection?p_html_source=0`) for details. Once the operating system is installed and working, you can then install Oracle Clusterware to create the cluster.

Oracle Clusterware is installed independently of the Oracle Database. Once Oracle Clusterware is installed, you can install ASM, the Oracle Database, or Oracle RAC on any of the nodes in the cluster.

See Also: Your platform-specific Oracle Database installation documentation

Overview of Oracle Clusterware Platform-Specific Software Components

When Oracle Clusterware operates, several platform-specific processes or services also run on each node in the cluster. The UNIX, Linux, and Windows processes are described in the following sections:

- Oracle Clusterware Processes on Linux and UNIX Systems
- Oracle Clusterware Services on Windows Systems

Oracle Clusterware Processes on Linux and UNIX Systems

Oracle Clusterware processes on Linux and UNIX systems include the following:

- `crsd`—Performs high availability recovery and management operations such as maintaining the OCR and managing application resources. This process runs as `LocalSystem`. This process restarts automatically upon failure.
- `evmd`—Event manager daemon. This process also starts the `racgevt` process to manage FAN server callouts.
- `ocssd`—Manages cluster node membership and runs as the `oracle` user; failure of this process results in a node restart.
- `oprocd`—Process monitor for the cluster. Note that this process only appears on platforms that do not use third-party vendor clusterware with Oracle Clusterware.

Note: Oracle Clusterware on Linux platforms can have multiple threads that appear as separate processes with separate process identifiers.

Oracle Clusterware Services on Windows Systems

Oracle Clusterware services on Windows systems include the following:

- `OracleCRService`—Performs high availability recovery and management operations such as maintaining the OCR and managing application resources. This process as the `LocalSystem` user on Windows. This process restarts automatically upon failure.
- `OracleCSService`—Manages cluster node membership and runs as the `oracle` user who installed Oracle Clusterware; failure of this process results in cluster restart.
- `OracleEVMService`—Event manager daemon. This process also starts the `racgevt` process to manage FAN server callouts.
- `OraFenceService`—Process monitor for the cluster.

Oracle Clusterware Subcomponent Processes and Background Processes

Oracle Clusterware comprises several processes that facilitate cluster operations. The Cluster Ready Services (CRS), Cluster Synchronization Service (CSS), Event Management (EVM), and Oracle Clusterware components communicate with other cluster component layers in the other instances in the same cluster database environment. These components are also the main communication links between the Oracle Database, applications, and the Oracle Clusterware high availability components. In addition, these background processes monitor and manage database operations.

> **See Also:** Chapter 5, "Making Applications Highly Available Using Oracle Clusterware" for more detailed information about the Oracle Clusterware API

The following list describes some of the major Oracle Clusterware background processes. The list includes components that are processes on Linux and UNIX operating systems, or services on Windows.

- **Cluster Ready Services (CRS)—**The primary program for managing high availability operations in a cluster. Anything that the `CRS` process manages is known as a cluster resource, which could be a database, an instance, a service, a listener, a virtual IP (VIP) address, an application process, and so on. The `CRS` process manages cluster resources based on the resource's configuration information that is stored in the OCR. This includes start, stop, monitor and failover operations. The `CRS` process generates events when a resource status changes. When you have installed Oracle RAC, the `CRS` process monitors the Oracle instance, listener, and so on, and automatically restarts these components when a failure occurs. By default, the `CRS` process makes three attempts to start the Oracle Notification Service (ONS), one attempt to start an Oracle Database, and five attempts to restart other database components. The `CRS` process does not attempt to restart the VIP. After these initial attempts, the `CRS` process does not make further restart attempts if the resource does not restart.
- **Cluster Synchronization Services (CSS)—**Manages the cluster configuration by controlling which nodes are members of the cluster and by notifying members when a node joins or leaves the cluster. If you are using third-party clusterware, then the `css` process interfaces with your clusterware to manage node membership information.

- **Event Management (EVM)**—A background process that publishes events that Oracle Clusterware creates.
- **Oracle Notification Service (ONS)**—A publish and subscribe service for communicating Fast Application Notification (FAN) events.
- **RACG—**Extends clusterware to support Oracle-specific requirements and complex resources. Runs server callout scripts when FAN events occur.
- **Process Monitor Daemon (OPROCD)**—This process is locked in memory to monitor the cluster and provide I/O fencing. The `OPROCD` periodically wakes up and checks that the interval since it last awoke is within the expected time. If not, then `OPROCD` resets the processor and restarts the node. An `OPROCD` failure results in Oracle Clusterware restarting the node.

In Table 1–1, if a UNIX or a Linux system process has an (`r`) beside it, then the process runs as the `root` user. If a Windows system service has an (A) beside it, then the service runs as the `Administrative` user. Otherwise the process or service runs as the `oracle` user.

Table 1–1 List of Processes and Services Associated with Oracle Clusterware

Oracle Clusterware Component	Linux/UNIX Process	Windows Services	Windows Processes
Process Monitor Daemon	`oprocd` (`r`)	`OraFenceService`	
RACG	`racgmain`, `racgimon`		`racgmain.exe` `racgimon.exe`
Oracle Notification Service (ONS)	`ons`		`ons.exe`
Event Manager	`evmd` (`r`), `evmd.bin`, `evmlogger`	`OracleEVMService`	`evmlogger.exe`, `evmd.exe`
Cluster Ready Services (CRS)	`crsd.bin` (`r`)	`OracleCRService` (`A`) `OracleCRSToken_`*`user`*	`crsd.exe`
Cluster Synchronization Services (CSS)	`init.cssd` (`r`), `ocssd` (`r`), `ocssd.bin`	`OracleCSService`	`ocssd.exe`

See Also: "Clusterware Log Files and the Unified Log Directory Structure" on page F-7 for information about the location of log files created for processes

Overview of Installing Oracle Clusterware

Install Oracle Clusterware with the Oracle Universal Installer (OUI).

The following sections introduce the installation processes for Oracle Clusterware:

- Oracle Clusterware Version Compatibility
- About the Oracle Clusterware Installation

Oracle Clusterware Version Compatibility

You can install different releases of Oracle Clusterware, ASM, and the Oracle Database software on your cluster. Follow these guidelines when installing different releases of software on your cluster:

- There can be only be one installation of Oracle Clusterware running in the cluster, and it must be installed into its own home (*CRS_home*). The release of Oracle Clusterware you use must be equal to, or higher than the ASM and Oracle RAC versions running in the cluster; you cannot install a version of Oracle RAC that was released after the version of Oracle Clusterware that you are running on the cluster. That is:
  - Oracle Clusterware 11*g* Release 1 (11.1) supports ASM Release 11.1, 10.2, and 10.1.
  - Oracle Clusterware 11*g* Release 1 (11.1) supports Oracle Database Release 11.1, 10.2, and 10.1.
  - ASM Release 11.1 requires Oracle Clusterware Release 11.1 and supports Oracle Database Release 11.1, 10.2, and 10.1.
  - Oracle Database Release 11.1 requires Oracle Clusterware Release 11.1 and (if you are using ASM storage) you can run different releases of Oracle Database and ASM.

    For example:

    * If you have Oracle Clusterware 11*g* Release 1 installed as your clusterware, then you can have an Oracle Database 10*g* Release 1 single-instance database running on one node, and separate Oracle RAC 10*g* Release 1, Release 2, and Oracle RAC 11*g* Release 1 databases also running on the cluster. However, you cannot have Oracle Clusterware 10*g* Release 2 installed on your cluster, and install Oracle RAC 11*g*. You can install Oracle Database 11*g* (single-instance) on a node in an Oracle Clusterware 10*g* Release 2 cluster.
    * When using different release ASM and Oracle Database releases, the functionality of each is dependent on the functionality of the earlier software release. Thus, if you install Oracle Clusterware 11*g* and you later install ASM, and you use it to support an existing Oracle Database 10*g* release 10.2.0.3 installation, then ASM functionality is equivalent only to that available in the 10.2 release version.
- There can be multiple Oracle homes for the Oracle Database (both single instance and Oracle RAC) in the cluster. Note that the Oracle RAC databases must be running Oracle Database 10*g* Release 1 (10.1) or higher.
- You can use different users for the Oracle Clusterware and Oracle Database homes as long as they belong to the same primary group.
- There can only be one installation of ASM running in the cluster. It is recommended that ASM is running the same (or higher) release than the release of the Oracle database.
- For Oracle RAC running Oracle9*i* you must run an Oracle9*i* cluster. For UNIX systems, that is HACMP, Serviceguard, Sun Cluster, or Veritas SF. For Windows and Linux systems, that is the Oracle Cluster Manager. If you want to install Oracle RAC 10*g*, then you must also install Oracle Clusterware.
- You cannot install Oracle9*i* RAC on an Oracle 10*g* cluster. If you have an Oracle9*i* RAC cluster, you can add Oracle RAC 10*g* and they will work together. However,

once you have installed Oracle Clusterware 10*g*, you can no longer install any new Oracle9*i* RAC.

- Oracle recommends that you do not run different cluster software on the same servers unless they are certified to work together. However, if you are adding Oracle RAC to servers that are part of a cluster, either migrate to Oracle Clusterware or ensure that:
  - The clusterware you are running is supported to run with Oracle RAC release 10*g*.
  - You have installed the correct options for Oracle Clusterware and the other-vendor clusterware to work together.

    See Also: Your platform-specific Oracle Clusterware installation guide for more version compatibility information

About the Oracle Clusterware Installation

This section discusses Oracle Clusterware installations at a high level. For detailed installation instructions, see your platform-specific Oracle Clusterware installation guide.

Oracle Clusterware is distributed on the Oracle Database installation media. The Oracle Universal Installer (OUI) installs Oracle Clusterware into a directory structure referred to as *CRS home*. This home is separate from the home directories for other Oracle products installed on the same server. OUI creates the Oracle Clusterware home directory for you. Before you start the installation, you must have sufficient disk space on a file system for the Oracle Clusterware directory. As a part of the installation and configuration, the CRS home and all of its parent directories are changed to be owned by the `root` user.

Because Oracle Clusterware works closely with the operating system, system administrator access is required for some of the installation tasks. In addition, some of the Oracle Clusterware processes must run as the system administrator, which is generally the `root` user on Linux and UNIX systems and the `LocalSystem` account on Windows systems.

Before you install Oracle Clusterware, Oracle recommends that you run the Cluster Verification Utility (CVU) to ensure that your environment meets the Oracle Clusterware installation requirements. The OUI also automatically runs CVU at the end of the clusterware installation to verify various clusterware components. The CVU simplifies the installation, configuration, and overall management of the Oracle Clusterware installation process by identifying problems in cluster environments.

During the Oracle Clusterware installation, you must identify three IP addresses for each node that is going to be part of your installation. One IP address is for the private interconnect, one is for the public interconnect, and the third IP address is the virtual IP address that clients will use to connect to each instance.

The Oracle Clusterware installation process creates the voting disk and OCR on cluster-aware storage. When you use normal redundancy, Oracle Clusterware maintains two copies of the OCR file and three copies of the voting disk file. This prevents the files from becoming single points of failure. Normal redundancy also eliminates the need for third-party storage redundancy solutions.

Note: If you choose external redundancy for the OCR and voting disk, then to enable redundancy, the disk subsystem must be configurable for RAID mirroring. Otherwise, your system may be vulnerable because the OCR and voting disk are single points of failure.

Overview of Managing Oracle Clusterware Environments

The following list describes the tools and utilities available to manage your Oracle Clusterware environment:

- **Oracle Enterprise Manager**—Enterprise Manager has both the Database Control and Grid Control GUI interfaces for managing both single instance and Oracle RAC database environments. Oracle recommends using Enterprise Manager to perform administrative tasks.

  See Also: *Oracle Database 2 Day + Real Application Clusters Guide* and *Oracle Real Application Clusters Administration and Deployment Guide* for more information about administering Oracle Clusterware with Enterprise Manager

- **Cluster Verification Utility (CVU)**—CVU is a command-line tool that you can use to verify a range of cluster and Oracle RAC specific components such as shared storage devices, networking configurations, system requirements, and Oracle Clusterware, as well as operating system groups and users.

  Install and use CVU before you install Oracle Clusterware to ensure your configuration meets the minimum installation requirements. Also, use CVU to verify your configuration after completing administrative tasks, such as node additions and node deletions. You can use CVU for preinstallation checks as well as for post-installation checks of your cluster environment. CVU is especially useful during preinstallation and during installation of Oracle Clusterware and Oracle RAC components.

  See Also: Your platform-specific Oracle Clusterware and Oracle RAC installation guide for information about how to manually install CVU, and Appendix A, "Cluster Verification Utility Reference" for more information about using CVU

- **Server Control (SRVCTL)**—SRVCTL is a command-line interface that you can use to manage Oracle Clusterware, such as changing the VIP interface or nodeapps, from a single system.

  See Also: Server Control Utility reference appendix in the *Oracle Real Application Clusters Administration and Deployment Guide*

- **Cluster Ready Services Control (CRSCTL)**—CRSCTL is a command-line tool that you can use to manually control Oracle Clusterware. You use `crsctl` commands to start and stop Oracle Clusterware. The `crsctl` command has many options that help you perform a number tasks, such as enabling online debugging and dynamically adding and removing voting disks.

  See Also: Chapter 2, "Administering Oracle Clusterware" for more information about the `crsctl` commands

- **Oracle Interface Configuration Tool (OIFCFG)**—OIFCFG is a command-line tool for both single-instance Oracle databases and Oracle RAC environments that you can use to allocate and deallocate network interfaces to components. You can also use OIFCFG to direct components to use specific network interfaces and to retrieve component configuration information.

  See Also: Appendix C, "Oracle Interface Configuration (OIFCFG) Command Reference"

- **OCR Configuration Tool (OCRCONFIG)**—OCRCONFIG is a command-line tool for OCR administration. You can also use the `OCRCHECK` and `OCRDUMP` utilities to troubleshoot configuration problems that affect the OCR.

  See Also: Chapter 2, "Administering Oracle Clusterware" for more information about managing the `OCR`

Overview of Extending or Removing Oracle Clusterware in Grid Environments

You can extend Oracle Clusterware in grid environments that have large numbers of nodes using cloned images of Oracle Clusterware homes. Oracle cloning is the preferred method of creating many new clusters by copying images of Oracle Clusterware software to other nodes that have similar hardware and software. Cloning is best suited to scenarios where you need to quickly create several clusters of the same configuration.

Oracle provides the following methods of extending Oracle Clusterware environments:

- Oracle cloning procedure
- Oracle Enterprise Manager cloning
- The `addNode.sh` script

For new installations or if you have to install only one cluster, then you should use the traditional automated and interactive installation methods, such as Oracle Universal Installer (OUI) or the Provisioning Pack feature of Oracle Enterprise Manager. If your goal is to add or delete Oracle Clusterware from nodes in the cluster, you can use the `addNode.sh` and `rootdelete.sh` scripts.

The cloning process assumes you successfully installed an Oracle Clusterware home on at least one node using the instructions in your platform-specific Oracle Clusterware installation guide. In addition, ensure that all root scripts run successfully on the node from which you are extending your cluster.

See Also:

- Chapter 3, "Cloning Oracle Clusterware" for step-by-step cloning procedures
- Oracle Enterprise Manager online Help system for more information about the Provisioning Pack
- Chapter 4, "Adding and Deleting Oracle Clusterware Homes"

Overview of the Oracle Clusterware High Availability Framework and Application Programming Interface

Oracle Clusterware provides a high availability application programming interface (API) that you use to enable Oracle Clusterware to manage applications or processes that run a cluster. The API enables you to provide high availability for all of your applications. Oracle Clusterware with ASM enables you to create a consolidated pool of storage to support both single-instance Oracle databases and the Oracle RAC databases that are running.

You can define a virtual IP address for an application so users can access the application independently of the node in the cluster where the application is running. This is referred to as the application VIP. You can define multiple application VIPs, with generally one application VIP defined for each application running. The application VIP is tied to the application by making it dependent on the application resource defined by Cluster Ready Services (CRS).

To maintain high availability, Oracle Clusterware components can respond to status changes to restart applications and processes according to defined high availability rules. You can use the Oracle Clusterware high availability framework by registering your applications with Oracle Clusterware and configuring the clusterware to start, stop, or relocate your application processes. That is, you can make custom applications highly available by using Oracle Clusterware to create profiles that monitor, relocate, and restart your applications.

For Oracle RAC to respond consistently and quickly to a failure, the virtual IP address removes network timeouts from the recovery process. When a node fails, its virtual IP relocates to another node in the cluster.

> **See Also:** Chapter 5, "Making Applications Highly Available Using Oracle Clusterware" for more detailed information about the Oracle Clusterware API

2

Administering Oracle Clusterware

This chapter describes how to administer the voting disks and the Oracle Cluster Registry (OCR) under the following topics:

- Administering Voting Disks
- Administering the Oracle Cluster Registry
- Changing Network Addresses

Administering Voting Disks

Oracle Clusterware includes two important components: the voting disk and the OCR.

- The voting disk is a file that manages information about node membership.
- The OCR is a file that manages cluster and Oracle Real Application Clusters (Oracle RAC) database configuration information.

Oracle recommends that you select the option to configure multiple voting disks during Oracle Clusterware installation to improve availability. If necessary, you can dynamically add voting disks after you complete the Oracle Clusterware installation process.

After installation, use the following procedures to regularly backup your voting disks and to recover them as needed:

- Backing Up Voting Disks
- Recovering Voting Disks
- Adding, Moving, or Deleting Voting Disks

Backing Up Voting Disks

Oracle recommends that you back up your voting disk after the initial cluster creation and after you complete any node addition or deletion procedures.

First, as `root` user, stop Oracle Clusterware (with the `crsctl stop crs` command) on all nodes. Then, determine the current voting disk by issuing the following command:

```
crsctl query votedisk css
```

Then, issue the `dd` or `ocopy` command to back up a voting disk, as appropriate. In the following examples, *`voting_disk_name`* is the name of the active voting disk and *`backup_file_name`* is the name of the file to which you want to back up the voting disk contents:

- On Linux or UNIX systems:

  ```
  dd if=voting_disk_name of=backup_file_name
  ```

  Oracle recommends you use the `dd` command to backup the voting disk with a minimum block size of 4KB.

- On Windows systems, use the `ocopy` command:

  ```
  ocopy voting_disk_name backup_file_name
  ```

  To display online documentation for `OCOPY`, enter `OCOPY` by itself at the Windows prompt.

Recovering Voting Disks

To restore the backup of your voting disk, issue the `dd` or `ocopy` command.

In the following examples, *`backup_file_name`* is the name of the voting disk backup file and *`voting_disk_name`* is the name of the active voting disk:

- On Linux or UNIX systems:

  ```
  dd if=backup_file_name of=voting_disk_name
  ```

  Oracle recommends you use the `dd` command to backup the voting disk with a minimum block size of 4KB.

- On Windows systems, use the `ocopy` command:

  ```
  ocopy backup_file_name voting_disk_name
  ```

If you have multiple voting disks, then you can remove the voting disks and add them back into your environment using the following commands (where *`path`* is the complete path of the location where the voting disk resides):

```
crsctl delete votedisk css path
crsctl add votedisk css path
```

Adding, Moving, or Deleting Voting Disks

You can add, move, and remove voting disks after Oracle Clusterware has been installed. Before making any modification to the voting disk, as `root` user, stop Oracle Clusterware using the `crsctl stop crs` command on all nodes. The following commands describe how to add, move, or remove voting disks:

- To retrieve the current voting disk, issue the following command:

  ```
  crsctl query css votedisk
  ```

  This command lists the voting disks used by Cluster Synchronization Services (CSS).

- To add a voting disk, issue the following command as the `root` user, replacing the `path` variable with the fully qualified path name for the voting disk you want to add:

  ```
  crsctl add votedisk css path -force
  ```

- To move a voting disk, issue the following commands as the `root` user, replacing the `path` variable with the fully qualified path name for the voting disk you want to move:

  ```
  crsctl delete votedisk css path -force
  ```

```
crsctl add votedisk css path -force
```

- To remove a voting disk, issue the following command as the `root` user, replacing the `path` variable with the fully qualified path name for the voting disk you want to remove:

```
crsctl delete votedisk css path -force
```

> **Note:** If your cluster is down, then you can include the `-force` option to modify the voting disk configuration, without interacting with active Oracle Clusterware daemons. However, using the `-force` option while any cluster node is active may corrupt your configuration.

After modifying the voting disk, restart Oracle Clusterware using the `crsctl start crs` command on all nodes, and verify the voting disk location using the following command:

```
crsctl query css votedisk css
```

Administering the Oracle Cluster Registry

This section describes how to administer the OCR using the OCR tools: OCRCONFIG, OCRDUMP, and OCRCHECK.

The OCR contains information about the cluster node list, instance-to-node mapping information, and information about Oracle Clusterware resource profiles for applications that you have customized as described in Chapter 5, "Making Applications Highly Available Using Oracle Clusterware".

This section describes how to administer the OCR in the following topics:

- Adding, Replacing, Repairing, and Removing the Oracle Cluster Registry
- Managing Backups and Recovering the Oracle Cluster Registry
- Diagnosing Oracle Cluster Registry Problems
- Overriding the Oracle Cluster Registry Data Loss Protection Mechanism
- Administering the Oracle Cluster Registry with OCR Export and Import Commands
- Implementing the Oracle HARD Initiative for the Oracle Cluster Registry
- Upgrading and Downgrading the Oracle Cluster Registry Configuration

> **See Also:** Appendix E, "Oracle Cluster Registry Configuration Tool Command Reference" for information about the OCRCONFIG tool, and Appendix F, "Troubleshooting Oracle Clusterware" for information about the OCRDUMP and OCRCHECK utilities

Adding, Replacing, Repairing, and Removing the Oracle Cluster Registry

The Oracle installation process for Oracle Clusterware gives you the option of automatically mirroring the OCR. You can put the mirrored OCR on an Oracle cluster file system disk, on a shared raw device, or on a shared raw logical volume. In addition, Oracle supports block devices for the OCR on Linux.

You can manually mirror the OCR, as described in the "Adding an Oracle Cluster Registry" section, if you:

- Upgraded to release 11.1 but did not choose to mirror the OCR during the upgrade
- Created only one OCR during the Oracle Clusterware installation

> **Note:** Oracle *strongly* recommends that you use mirrored OCRs if the underlying storage is not RAID. Mirroring can help to prevent the OCR from becoming a single point of failure.

In addition to mirroring the OCR, you can also:

- Replace the OCR if Oracle displays an OCR failure alert in Enterprise Manager or in the Oracle Clusterware alert log file, as described in the "Replacing an Oracle Cluster Registry" section.
- Repair an OCR location if there is a misconfiguration or other type of OCR error, as described in the "Repairing an Oracle Cluster Registry Configuration on a Local Node" section.
- Remove an OCR location if, for example, your system experiences a performance degradation due to OCR processing or if you transfer your OCR to RAID storage devices and chose to no longer use multiple OCRs. This is described in the "Removing an Oracle Cluster Registry" section.

> **Note:** The operations in this section affect the OCR clusterwide: they change the OCR configuration information in the `ocr.loc` file on Linux and UNIX systems and the Registry keys on Windows systems. However, the `ocrconfig` command cannot modify OCR configuration information for nodes that are shut down or for nodes on which Oracle Clusterware is not running.

Adding an Oracle Cluster Registry

You can add an OCR location after an upgrade or after completing the Oracle Clusterware installation. If you already mirror the OCR, then you do not need to add an OCR location; Oracle automatically manages two OCRs when it mirrors the OCR. Oracle Clusterware environments do not support more than two OCRs: a primary OCR and a secondary OCR.

> **Note:** If the OCR resides on a cluster file system file or if the OCR is on a network file system, then create a dummy file before performing the procedures in this section.

As the `root` user, issue the following command to add an OCR location using either *`destination_file`* or *`disk`* to designate the target location of the additional OCR:

```
ocrconfig -replace ocr destination_file or disk
```

Run the following command to add an OCR mirror location using either *`destination_file`* or *`disk`* to designate the target location of the additional OCR:

```
ocrconfig -replace ocrmirror destination_file or disk
```

Note: On Linux and UNIX systems, you must be `root` user to run `ocrconfig` commands. On Windows systems, the user must be a member of the Administrator's group.

Replacing an Oracle Cluster Registry

If you need to change the location of an existing OCR, or change the location of a failed OCR to the location of a working one, you can use the following procedure as long as one OCR file remains online.

To change the location of an OCR:

1. Use the OCRCHECK utility to verify that a copy of the OCR other than the one you are going to replace is *online*, using the following command:

   ```
   ocrcheck
   ```

   OCRCHECK displays all OCR files that are registered and whether or not they are available (online). If an OCR file suddenly becomes unavailable, it might take a short period of time for Oracle Clusterware to show the change in status.

   Note: The OCR that you are *replacing* can be either online or offline.

2. Use the following command to verify that Oracle Clusterware is running on the node on which the you are going to perform the replace operation:

   ```
   crsctl check crs
   ```

3. Issue the following command as `root` user to replace the primary OCR using either *`destination_file`* or *`disk`* to indicate the target OCR location:

   ```
   ocrconfig -replace ocr destination_file
   ocrconfig -replace ocr disk
   ```

4. Issue the following command as `root` user to replace a secondary OCR using either *`destination_file`* or *`disk`* to indicate the target OCR location:

   ```
   ocrconfig -replace ocrmirror destination_file
   ocrconfig -replace ocrmirror disk
   ```

5. If any node that is part of your current Oracle RAC cluster is shut down, then run the following command on the stopped node to let that node rejoin the cluster after the node is restarted:

   ```
   ocrconfig -repair
   ```

Repairing an Oracle Cluster Registry Configuration on a Local Node

You may need to repair an OCR configuration on a particular node if your OCR configuration changes while that node is stopped. For example, you may need to repair the OCR on a node that was not up while you were adding, replacing, or removing an OCR. To repair an OCR configuration, run the following command on the node on which you have stopped the Oracle Clusterware daemon:

```
ocrconfig -repair ocrmirror device_name
```

This operation only changes the OCR configuration on the node from which you run this command. For example, if the OCR mirror device name is `/dev/raw1`, then use

the command syntax `ocrconfig -repair ocrmirror /dev/raw1` on this node to repair its OCR configuration.

> **Note:** You *cannot* perform this operation on a node on which the Oracle Clusterware daemon is running.

Removing an Oracle Cluster Registry

To remove an OCR location, at least one other OCR must be online. You can remove an OCR location to reduce OCR-related overhead or to stop mirroring your OCR because you moved the OCR to redundant storage such as RAID.

Perform the following procedure as the `root` user to remove an OCR location from your Oracle Clusterware environment:

1. Ensure that *at least one* OCR other than the OCR that you are removing is online.

   > **Caution:** Do *not* perform this OCR removal procedure unless there is at least one other active OCR online.

2. Issue the following command on any node in the cluster to remove the OCR device:

   ```
   ocrconfig -replace ocr
   ```

   Issue the following command on any node in the cluster to remove the mirrored OCR device:

   ```
   ocrconfig -replace ocrmirror
   ```

   These commands update the OCR configuration on all of the nodes on which Oracle Clusterware is running.

   > **Note:** When removing an OCR location, the remaining OCR must be online. If you remove a primary OCR, then the mirrored OCR becomes the primary OCR.

Overriding the Oracle Cluster Registry Data Loss Protection Mechanism

The OCR has a mechanism that prevents data loss due to accidental overwrites. If you configure a mirrored OCR and if Oracle Clusterware cannot access the two mirrored OCR locations and also cannot verify that the available OCR location contains the most recent configuration, then Oracle Clusterware prevents further modification to the available OCR location. In addition, the process prevents overwriting by prohibiting Oracle Clusterware from starting on the node on which only one OCR is available. In such cases, Oracle displays an alert message in either Oracle Enterprise Manager, the Oracle Clusterware alert log files, or both. If this problem is local to only one node, you can use other nodes to start your cluster database.

However, if you are unable to start any cluster node in your environment and if you can neither repair the OCR nor restore access to all OCR locations, then you can override the protection mechanism. The procedure described in the following list enables you to start the cluster using the available OCR location. However, overriding the protection mechanism can result in the loss of data that was not available at the time that the previous known good state was created.

> **Note:** Overriding the OCR using the following procedure can result in the loss of OCR updates that were made between the time of the last known good OCR update made to the currently accessible OCR and the time at which you performed the overwrite. In other words, running the `ocrconfig -overwrite` command can result in data loss if the OCR location that you are using to perform the overwrite does not contain the latest configuration updates for your cluster environment.

Perform the following procedure to overwrite the OCR if a node cannot start up *and* if the alert log contains CLSD-1009 and CLSD-1011 messages.

1. Attempt to resolve the cause of the CLSD-1009 and CLSD-1011 messages.

   Compare the node's OCR configuration (`ocr.loc` on Linux and UNIX systems and the Registry on Windows systems) with other nodes on which Oracle Clusterware is running.

   - If the configurations do not match, then run `ocrconfig -repair`.
   - If the configurations match, then ensure that the node can access all of the configured OCRs by running an `ls` command on Linux and UNIX systems. On windows, use a `dir` command if the OCR location is a file and run `GuiOracleObjectManager.exe` to verify that the partition with the name exists.

2. Ensure that the most recent OCR contains the latest OCR updates.

   Look at output from the `ocrdump` command and determine whether it has your latest updates.

3. If you cannot resolve the problem that caused the CLSD message, then run the command `ocrconfig -overwrite` to bring up the node.

Managing Backups and Recovering the Oracle Cluster Registry

This section describes how to backup OCR content and use it for recovery. The first method uses automatically generated OCR file copies and the second method enables you to issue a backup command on demand:

- **Automatic backups**—Oracle Clusterware automatically creates OCR backups every four hours. At any one time, Oracle always retains the last three backup copies of the OCR. The CRSD process that creates the backups also creates and retains an OCR backup for each *full day* and *at the end of each week*. You cannot customize the backup frequencies or the number of files that Oracle retains.
- **Manual backups**—Use the `ocrconfig -manualbackup` command to force Oracle Clusterware to perform a backup of the OCR at any time, rather than wait for the automatic backup that occurs at 4-hour intervals. The `-manualbackup` option is especially useful when you to need to obtain a binary backup on demand, such as before you make changes to the OCR.

You can use any backup software to copy the automatically generated backup files at least once daily to a different device from where the primary OCR resides.

The default location for generating backups on Linux or UNIX systems is *`CRS_home`*`/cdata/`*`cluster_name`* where *`cluster_name`* is the name of your cluster. The Windows default location for generating backups uses the same path structure. Because the default backup is on a local file system, Oracle recommends that you

include the backup file created with the ocrconfig command as part of your operating system backup using standard operating system or third-party tools.

> **See Also:** You can use the `ocrconfig -backuploc` option to move the location where OCR creates backups. Appendix E, "Oracle Cluster Registry Configuration Tool Command Reference" describes the OCR configuration tool options.

You can issue the `ocrconfig -showbackup` command to display the backup location, timestamp, and the originating node name of the backup files that Oracle created. By default, the `-showbackup` option displays information for both automatic and manual backups but you can include the `auto` or `manual` flag to display only the automatic backup information or only the manual backup information, respectively.

> **Note:** On Linux and UNIX systems, you must be `root` user to run `ocrconfig` commands. On Windows systems, the user must be a member of the Administrator's group.

> **See Also:** "Administering the Oracle Cluster Registry with OCR Export and Import Commands" on page 2-10 to use manually created OCR export files to copy OCR content and use it for recovery

Overview of Restoring the Oracle Cluster Registry from OCR Backups

If an application fails, then before attempting to restore the OCR, restart the application. As a definitive verification that the OCR failed, run an `ocrcheck` and if the command returns a failure message, then both the primary OCR and the OCR mirror have failed. Attempt to correct the problem using one of the following platform-specific OCR restoration procedures.

> **Note:** You *cannot* restore your configuration from an OCR backup file using the `-import` option, which is explained in "Administering the Oracle Cluster Registry with OCR Export and Import Commands" on page 2-10. You *must instead* use the `-restore` option, as described in the following sections.

- Restoring the Oracle Cluster Registry on Linux or UNIX Systems
- Restoring the Oracle Cluster Registry on Windows Systems

Restoring the Oracle Cluster Registry on Linux or UNIX Systems

Use the following procedure to restore the OCR on Linux or UNIX systems:

1. Identify the OCR backups using the `ocrconfig -showbackup` command. Review the contents of the backup using `ocrdump -backupfile` *`file_name`* where *`file_name`* is the name of the backup file.
2. Stop Oracle Clusterware on all of the nodes by executing the `crsctl stop crs` command on all of the nodes.
3. Perform the restore by applying an OCR backup file that you identified in Step 1 using the following command where *`file_name`* is the name of the OCR that you want to restore. Make sure that the OCR devices that you specify in the OCR configuration exist and that these OCR devices are valid before running this command.

```
ocrconfig -restore file_name
```

4. Restart Oracle Clusterware on all of the nodes in your cluster by restarting each node or by running the `crsctl start crs` command.
5. Run the following command to verify the OCR integrity where the `-n node_list all` argument retrieves a listing of all of the cluster nodes that are configured as part of your cluster:

```
cluvfy comp ocr [-n node_list] [-verbose]
```

> **See Also:** Appendix F, "Troubleshooting Oracle Clusterware" for more information about enabling and using CVU

Restoring the Oracle Cluster Registry on Windows Systems

Use the following procedure to restore the OCR on Windows systems:

1. Identify the OCR backups using the `ocrconfig -showbackup` command. Review the contents of the backup using `ocrdump -backupfile file_name` where `file_name` is the name of the backup file.
2. On all of the remaining nodes, disable the following OCR clients and stop them using the Service Control Panel: `OracleClusterVolumeService`, `OracleCSService`, `OracleCRService`, and the `OracleEVMService`.
3. Execute the restore by applying an OCR backup file that you identified in Step 1 with the `ocrconfig -restore file name` command. Make sure that the OCR devices that you specify in the OCR configuration exist and that these OCR devices are valid.
4. Start all of the services that were stopped in step 2. Restart all of the nodes and resume operations in cluster mode.
5. Run the following command to verify the OCR integrity where the `-n node_list all` argument retrieves a listing of all of the cluster nodes that are configured as part of your cluster:

```
cluvfy comp ocr [-n node_list] [-verbose]
```

> **See Also:** Appendix A for more information about enabling and using CVU

Diagnosing Oracle Cluster Registry Problems

You can use the OCRDUMP and OCRCHECK utilities to diagnose OCR problems as described under the following topics:

- Using the OCRDUMP Utility
- Using the OCRCHECK Utility

Using the OCRDUMP Utility

Use the OCRDUMP utility to write the OCR contents to a file so that you can examine the OCR content.

> **See Also:** "OCRDUMP Utility Syntax and Options" on page F-9 for more information about the OCRDUMP utility

Using the OCRCHECK Utility

Use the OCRCHECK utility to verify the OCR integrity.

See Also: "Using the OCRCHECK Utility" on page F-10 for more information about the OCRCHECK utility

Administering the Oracle Cluster Registry with OCR Export and Import Commands

In addition to using the automatically created OCR backup files, you should also export the OCR contents before and after making significant configuration changes, such as adding or deleting nodes from your environment, modifying Oracle Clusterware resources, or creating a database. Do this by using the `ocrconfig -export` command. This exports the OCR content to a file format.

Using the `ocrconfig -export` command enables you to restore the OCR using the `-import` option if your configuration changes cause errors. For example, if you have unresolvable configuration problems, or if you are unable to restart your clusterware after such changes, then restore your configuration using one of the following platform-specific procedures:

- Importing Oracle Cluster Registry Content on Linux or UNIX Systems
- Importing Oracle Cluster Registry Content on Windows Systems

Note: Most configuration changes that you make not only change the OCR contents, configuration changes also cause file and database object creation. Some of these changes are often not restored when you restore the OCR. Do not perform an OCR restore as a correction to revert to previous configurations if some of these configuration changes should fail. This may result in an OCR that has contents that do not match the state of the rest of your system.

Importing Oracle Cluster Registry Content on Linux or UNIX Systems

Use the following procedure to import the OCR on Linux or UNIX systems:

1. Stop Oracle Clusterware by issuing the following command (as `root` user) on all of the nodes:

   ```
   crsctl stop crs
   ```

2. Import the OCR by applying an OCR export file by issuing the following command (as `root` user), where *`file_name`* is the name of the OCR file from which you want to import OCR information:

   ```
   ocrconfig -import file_name
   ```

3. Restart Oracle Clusterware on all of the nodes in your cluster:

   ```
   crsctl start crs
   ```

4. Verify the OCR integrity by issuing the following Cluster Verification Utility (CVU) command, where the *`-n node_list all`* argument retrieves a listing of all of the cluster nodes that are configured as part of your cluster:

   ```
   cluvfy comp ocr [-n node_list] [-verbose]
   ```

Note: You *cannot* import an exported OCR backup file. (Managing backups is described in "Managing Backups and Recovering the Oracle Cluster Registry" on page 2-7.) You must instead use the `-import` option as described in the following sections.

See Also: Appendix F, "Troubleshooting Oracle Clusterware" for more information about enabling and using CVU

Importing Oracle Cluster Registry Content on Windows Systems

Use the following procedure to import the OCR on Windows systems:

1. Identify the OCR export file that you want to import by running the `ocrconfig -showbackup` command.
2. Stop the following OCR clients *on each node* in your Oracle Clusterware environment using the Service Control Panel: `OracleClusterVolumeService`, `OracleCMService`, `OracleEVMService`, `OracleCSService`, and `OracleCRService`.
3. Import an OCR export file using the `ocrconfig -import` command from one node.
4. Restart all of the affected services on all of the nodes.
5. Run the following Cluster Verification Utility (CVU) command to verify the OCR integrity where *`node_list`* is a list of all of the nodes in your cluster database:

```
cluvfy comp ocr [-n node_list] [-verbose]
```

See Also: Appendix F, "Troubleshooting Oracle Clusterware" for more information about enabling and using CVU

Implementing the Oracle HARD Initiative for the Oracle Cluster Registry

The Oracle Hardware Assisted Resilient Data (HARD) initiative prevents data corruptions from being written to permanent storage. If you enable HARD, then the OCR writes HARD-compatible blocks. To determine whether the device used by the OCR supports HARD and then enable it, review the Oracle HARD white paper at:

`http://www.oracle.com/technology/deploy/availability/htdocs/HARD.html`

Upgrading and Downgrading the Oracle Cluster Registry Configuration

When you install Oracle Clusterware, Oracle automatically runs the `ocrconfig -upgrade` command. To downgrade, follow the downgrade instructions for each component and also downgrade the OCR using the `ocrconfig -downgrade` command. If you are upgrading the OCR, then you can use the `ocrcheck` command to verify the integrity of the OCR.

Changing Network Addresses

In an Oracle Clusterware configuration, there must be a minimum of two network connections. One connection is for the public network interface on which users and application servers connect to access data on the database server, and the other connection is for the private network interface between the nodes in the cluster.

This section contains the following topics:

- Changing Public Network Addresses
- Changing Private Network Addresses

Changing Public Network Addresses

Clients use the Virtual Internet Protocol (VIP) address to connect to the Oracle RAC database instances. The VIP address resolves all timeouts entering the TCP/IP stack, resulting in higher application availability. The VIP address is a static IP address that defines and resolves to a hostname through the Domain Name System (DNS).

During the installation of Oracle Clusterware, you are prompted to enter a VIP address and a virtual hostname for each node in the cluster. These items are stored in the OCR and different components in the high availability framework depend on the VIP addresses.

Changing the VIP address requires modification of the node applications, including the VIP address, Global Services Daemon (GSD), the listener, and Oracle Notification Services (ONS). You can modify the VIP address while the node applications are running. However, the changes do not take effect until you restart the VIP address (hence, restarting the node applications). Other resources on a node, such as database instances and ASM instances, are dependent on the VIP address. Therefore, stopping the node applications takes the VIP address offline only if you stop the other resources.

Note: The following instructions describe how to change only a VIP address and assume that the hostname associated with the VIP address will not change.

- If you are changing only the VIP address, then it is not necessary to make changes to the `listener.ora` and `tnsnames.ora` files, provided they are using the VIP hostnames for the `HOST=` entries.
- If you are changing only the VIP address, then update the Domain Name System (DNS) or the hosts of the clients. Also, update the hosts entries of the server, too, if those are used for VIPs.
- If you are changing both the VIP hostname and the VIP address for a node, then you must modify the `listener.ora` file and change the `HOST=` entries to the new VIP hostname.

  You can use an editing tool to modify the `listener.ora` file manually or use the Net Configuration Assistant (NETCA) to remove the old listener and create a new listener. In addition, you do not need to make changes to the `tnsnames.ora` file of any clients connecting to the old hostname.

Perform the following steps to change a VIP address:

1. Stop all database and ASM instances. Stop all resources that are dependent on the VIP address on a given node. This includes all Oracle RAC database instances on that node, and the ASM instance on that node, if it exists:

   a. Stop database instances:

   ```
   srvctl stop instance -d grid -i grid1
   ```

   This example specifies the database name (`grid`) using the `-d` option and specifies the instance (`grid1`) on the appropriate node using the `-i` option.

   Alternatively, to stop the entire database, on all nodes, you can issue the `stop database` command, as follows:

   ```
   srvctl stop database -d grid
   ```

   b. To stop the ASM instance on the node, issue the following command:

```
srvctl stop asm -n mynode
```

Note: Alternatively, you can stop these resources using SQL*Plus statements, or on Windows systems you can stop the associated services.

2. Confirm the current IP address for the VIP by issuing the `ifconfig -a` command. This command displays the current VIP bound to one of the network interfaces. The following example displays the current VIP address that is bound to one of the network interfaces (`eth0:1`):

```
eth0:1    Link encap:Ethernet  HWaddr 00:01:03:2C:69:BB
          inet addr:192.168.1.125  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          ...
          ...
```

Note: On Windows systems, issue the `ipconfig /all` command, to list all IP addresses bound to a given adapter, including the VIP address.

3. Stop node applications using the `srvctl stop nodeapps` command. For example:

```
srvctl stop nodeapps -n mynode
```

This command stops the VIP address, the GSD, the listener, and the ONS daemon currently running on the specified node.

4. Verify that the VIP address is no longer running by issuing the `ifconfig -a` command on Linux and UNIX systems (or issue the `ipconfig /all` command on Windows systems), and confirm that the interface (in the example it was `eth0:1`) is no longer listed in the output.

   If the interface is still online, it indicates that a resource, which is dependent on the VIP address, is still running. Use the `crs_stat` command to show resources that are still online.

5. Make any changes necessary to the `/etc/hosts` files on all nodes on Linux and Unix systems, or the `%SystemRoot%\system32\drivers\etc\hosts` file on Windows systems, and make any necessary DNS changes to associate the new IP address with the old host name.

6. Modify the node applications and provide the new VIP address using the following `srvctl modify nodeapps` command syntax:

```
srvctl modify nodeapps -n node_name; [-o oracle_home] [-A new_vip_address]
```

   In the syntax statement include the following options:

   - `-n` *`node_name`* is the node name
   - `-o` *`Oracle_home`* is the Oracle home for the cluster database.
   - `-A` *`new_vip_address`* is the node-level VIP address: *`name`*|*`ip`*`/netmask[/if1[|if2|...]])`

For example, issue the following command as the `root` user:

```
srvctl modify nodeapps -n mynode -A 192.168.2.125/255.255.255.0/eth0
```

Attempting to issue this command as the `oracle` user results in the `PRKO-2117: This command should be executed as the system privilege user` error

7. Start node-level applications by issuing the `srvctl start nodeapps` command:

   ```
   srvctl start nodeapps -n node_name
   ```

   The following command example starts applications on the node named `mynode`:

   ```
   srvctl start nodeapps -n mynode
   ```

8. Repeat the steps for each node in the cluster.

   Because the SRVCTL utility is a clusterwide management tool, you can accomplish these tasks for any specific node from any node in the cluster, without the need to log in to each of the cluster nodes.

9. Run the following command to verify node connectivity between all of the nodes for which your cluster is configured. This command discovers all of the network interfaces available on the cluster nodes and verifies the connectivity between all of the nodes by way of the discovered interfaces. This command also lists all of the interfaces available on the nodes which are suitable for use as VIP addresses.

   ```
   cluvfy comp nodecon -n node_list all [-verbose]
   ```

Changing Private Network Addresses

Oracle Clusterware requires that each node is connected through a private network (in addition to the public network). The private network connection is referred to as the *cluster interconnect*. During the installation of Oracle Clusterware, you are prompted to enter a private network IP address for each node in the cluster. This address—unlike the VIP addresses—must be assigned already to a certain network interface on each node.

In addition, you must provide a private node name, which resolves to the IP address given, and is usually stored in the hosts file of the operating system. This private node name is stored automatically in the OCR. Oracle does not recommend using Domain Name Services (DNS) to resolve private network IP address names.

Oracle currently only supports clusters in which all nodes use the same network interface connected to the same subnet (defined as a global interface with the `oifcfg` command). You cannot use different network interfaces for each node (node-specific interfaces). See Appendix C, "Oracle Interface Configuration (OIFCFG) Command Reference" for more information about global and node-specific interfaces.

Table 2–1 describes how the Network Interface Card (NIC), the private IP address, and the private node name are stored.

Table 2–1 Storage for the NIC, Private IP Address, and Private Host Name

Entity	Stored in ...	Comments
Network Interface Card (NIC)	Operating system For example: `eth1`	Must be the same on all nodes. It can be changed globally.

Table 2–1 (Cont.) Storage for the NIC, Private IP Address, and Private Host Name

Entity	Stored in ...	Comments
Private network IP address	Operating system; assigned to the network interface. For example: `eth1` 10.10.1.1	Configure this as `root` using the `ifconfig` command. You can change the IP address locally if it is in the same subnet. Otherwise, change this globally.
Private node name	The `/etc/hosts` file, which resolves the private IP address by name	You cannot use the steps described in this section to change the private node name, which is also stored in the OCR, after Oracle Clusterware is installed.

The following topics describe how to change a private network address, the network interface, or both.

> **Note:** You cannot use the steps in this section to change the private node name after you have installed Oracle Clusterware.

To change a private network IP address:

If you want to change a private network IP address on one node and the new IP address is on the same subnet as the former IP address, perform the following steps:

1. Stop Oracle Clusterware by issuing the following command as `root` user on the node on which you want to change the IP address:

   ```
   crsctl stop crs
   ```

2. Assign a new network address to the Network Interface Card (NIC).[1]

   As `root` user, assign a new network address to the NIC using the `ifconfig` operating system command. Ensure the new private IP address is in the same subnet as the private IP addresses of the remaining nodes in the cluster.

3. Update the hosts file.

   The private network address is a static IP address. Any changes that you make to the private network address must be reflected in the hosts file of the operating system.

   > **Note:** Oracle does not recommend that you use the Domain Name System (DNS) for the private node name resolution. However, if you use DNS, you must change the DNS entries accordingly.

4. Restart Oracle Clusterware by issuing the following command as `root` user

   ```
   crsctl start crs
   ```

   The changes take effect when Oracle Clusterware restarts. To change more than one private IP address on more than one node in the cluster, repeat steps 1 through 3 in a rolling manner for each of those nodes.

[1] Because private network addresses are physical addresses, the system administrator must issue an `ifconfig` command to assign the new private network address to the NIC. See your platform-specific operating system documentation for more information about issuing the `ifconfig` command.

To change a private network IP address so that a new subnet is used:

If you want to change a private network IP address so that a new subnet is used, you must change all IP addresses on all cluster nodes to reflect the change of the IP subnet. Perform the following steps to change the subnet:

1. On all nodes, stop Oracle Clusterware by issuing the following command as `root` user:

   ```
   crsctl stop crs
   ```

2. Assign a new network address from the new subnet to the NIC.[1]

   As `root` user, assign a new network address to the NIC using the `ifconfig` operating system command. Ensure all private IP addresses are taken from the same subnet.

3. Restart Oracle Clusterware by issuing the following command as `root` user

   ```
   crsctl start crs
   ```

   The changes take effect when Oracle Clusterware restarts.

4. Issue the `oifcfg` command to change and store the new private IP address subnet in the OCR. See Appendix C for more information about using the OIFCFG command. The IP address changes take effect when Oracle Clusterware restarts.

Note: To minimize downtime, you can perform these steps, as follows:

1. Perform step 1 (to stop Oracle Clusterware) on all nodes in the cluster except one.
2. Perform steps 2 and 3 to change the IP address on each stopped node, but complete the steps on one node before performing the steps on the next node.
3. Perform step 1 to stop Oracle Clusterware on the running (unchanged) node.
4. Restart the nodes on which you have already changed the IP address.
5. Perform steps 2 and 3 to change the IP address on the last unchanged node.
6. Restart the last node.

If you use the `CLUSTER_INTERCONNECTS` initialization parameter, you must update it to reflect the changes. If you do not use the `CLUSTER_INTERCONNECT` parameter, then skip this step.

Note: Oracle does not recommend setting the `CLUSTER_INTERCONNECTS` parameter.

The private network address is a static IP address. Any changes that you make to the private network address must be reflected in the hosts file of the operating system.

Note: Oracle does not recommend that you use the Domain Name System (DNS) for the private node name resolution. However, if you use DNS, you must change the DNS entries accordingly.

To change the Network Interface Card (NIC) for the interconnect:

If you want to change the network interface for the private interconnect (for example, `eth1`), you must perform the change on all nodes (globally). This is because Oracle currently does not support the use of different network interface cards in the same subnet for the cluster interconnect.

Perform the following steps:

1. On all nodes, stop Oracle Clusterware by issuing the following command as `root` user:

   ```
   crsctl stop crs
   ```

2. Assign the current network address to the new NIC using the `ifconfig` command.

   As `root` user, issue the `ifconfig` operating system command to assign the currently used private network address to the NIC intended to be used for the interconnect. This usually requires some downtime for the current interface and the new interface. See your platform-specific operating system documentation for more information about issuing the `ifconfig` command.

3. Restart Oracle Clusterware by issuing the following command as `root` user on all nodes:

   ```
   crsctl start crs
   ```

4. Issue the `oifcfg` command to change and store the new private network/private IP address subnet assignment in the OCR. See Appendix C for more information about using the OIFCFG command. The changes take effect when Oracle Clusterware restarts.

Note: To minimize downtime, you can perform these steps as follows:

1. Perform step 1 (to stop Oracle Clusterware) on all nodes in the cluster except one.
2. Perform steps 2 and 3 to change the IP address on each stopped node, but complete the steps on one node before performing the steps on the next node.
3. Perform step 1 to stop Oracle Clusterware on the running (unchanged) node.
4. Restart the nodes on which you have already changed the IP address.
5. Perform steps 2 and 3 to change the IP address on the last unchanged node.
6. Restart the last node.

If you want to change both the private IP address and the network interface card in one step, assign the new IP address to the new NIC and then perform the remaining steps accordingly.

3

Cloning Oracle Clusterware

This chapter describes how to clone an existing Oracle Clusterware home and use it to create a new cluster or to extend Oracle Clusterware to new nodes on the same cluster. You implement cloning through the use of scripts in silent mode.

The cloning procedures described in this chapter are applicable to Linux, UNIX, and Windows systems. Although the examples in this chapter use Linux and UNIX commands, the cloning concepts and procedures apply to all platforms. For the Windows platform, you need to adjust the examples or commands to be Windows specific.

This chapter contains the following topics:

Introduction to Cloning Oracle Clusterware

Cloning is the process of copying an existing Oracle installation to a different location and then updating the copied installation to work in the new environment. The changes made by one-off patches applied on the source Oracle home are also present after the clone operation. During cloning, you run a script that replays the actions that installed the Oracle Clusterware home.

Cloning requires that you start with a successfully installed Oracle Clusterware home that you use as the basis for implementing a script that extends the Oracle Clusterware home to either create a new cluster or to extend the Oracle Clusterware environment to more nodes in the same cluster. Manually creating the cloning script can be prone to errors, because you must prepare the script without the benefit of any interactive checks to validate your input. Despite this, the initial effort is worthwhile for scenarios where you run a single script to install tens or even hundreds of clusters. If you have only one cluster to install, then you should use the traditional automated and interactive installation methods, such as Oracle Universal Installer (OUI) or the Provisioning Pack feature of Oracle Enterprise Manager.

> **Note:** Cloning is not a replacement for Oracle Enterprise Manager cloning that is a part of the Provisioning Pack. During Enterprise Manager cloning, the provisioning process simplifies the process by interactively asking you the details about the Oracle home (such as the location to which you want to deploy the clone, the name of the Oracle Database home, a list of the nodes in the cluster, and so on).
>
> The Provisioning Pack feature of Oracle Grid Control provides a framework that automates the provisioning of new nodes and clusters. For data centers with many clusters, the investment in creating a cloning procedure to provision new clusters and new nodes to existing clusters is worth the effort.

The following list describes some situations in which cloning is useful:

- Cloning provides a way to prepare a Oracle Clusterware home once and deploy it to many hosts simultaneously. You can complete the installation in silent mode, as a noninteractive process. You do not need to use a graphical user interface (GUI) console, and you can perform cloning from a Secure Shell (SSH) terminal session, if required.
- Cloning enables you to create a new installation (copy of a production, test, or development installation) with all patches applied to it in a single step. Once you have performed the base installation and applied all patch sets and patches on the source system, the clone performs all of these individual steps as a single procedure. This is in contrast to going through the installation process to perform the separate steps to install, configure, and patch the installation on each node in the cluster.
- Installing Oracle Clusterware by cloning is a quick process. For example, cloning an Oracle Clusterware home to a new cluster with more than two nodes requires a few minutes to install the Oracle software, plus a few minutes more for each node (approximately the amount of time it takes to run the `root.sh` script).
- Cloning provides a guaranteed method of repeating the same Oracle Clusterware installation on multiple clusters.

The cloned installation acts the same as the source installation. For example, you can remove the cloned Oracle Clusterware home using OUI or patch it using OPatch. You can also use the cloned Oracle Clusterware home as the source for another cloning operation. You can create a cloned copy of a test, development, or production installation by using the command-line cloning scripts. The default cloning procedure is adequate for most cases. However, you can also customize some aspects of the cloning process, for example, to specify custom port assignments or to preserve custom settings.

The cloning process works by copying all of the files from the source Oracle Clusterware home to the destination Oracle Clusterware home. Thus, any files used by the source instance that are located outside the source Oracle Clusterware home's directory structure are not copied to the destination location.

The size of the binary files at the source and the destination may differ because these are relinked as part of the cloning operation, and the operating system patch levels may also differ between these two locations. Additionally, the number of files in the cloned home would increase because several files copied from the source, specifically those being instantiated, are backed up as part of the clone operation.

Preparing the Oracle Clusterware Home for Cloning

To prepare the source Oracle Clusterware home to be cloned, you create a copy of an installed Oracle Clusterware home that you then use to perform the cloning procedure on one or more nodes.

Use the following step-by-step procedure to prepare a copy of the Oracle Clusterware home.

Step 1 Install Oracle Clusterware.

Use the detailed instructions in your platform-specific Oracle Clusterware installation guide to perform the following steps on the source node:

1. Install the Oracle Clusterware 11*g* release.
2. Install any patches that are required (for example, 11.1.0.*n*), if necessary.
3. Apply one-off patches, if necessary.

Step 2 Shutdown Oracle Clusterware.

Before copying the source Oracle Clusterware home, shut down Oracle Clusterware using the `crsctl stop crs` command. The following example shows the command and the messages that display during the shutdown:

```
[root@node1 root]# crsctl stop crs
Stopping resources.
This could take several minutes.
Successfully stopped Oracle Clusterware resources
Stopping Cluster Synchronization Services.
Shutting down the Cluster Synchronization Services daemon.
Shutdown request successfully issued.
```

Note that you copy the Oracle Clusterware home from only one of the nodes.

Step 3 Make a copy of the Oracle Clusterware home

To keep the installed Oracle Clusterware home as a working home, you should make a full copy of the source Oracle Clusterware home and remove the unnecessary files from the copy. For example, as the `root` user on Linux systems you could issue the `cp` command:

```
cp -prf CRS_HOME location_for_the_copy_of_crs_home
```

Step 4 Remove unnecessary files from the copy of the Oracle Clusterware home.

The Oracle Clusterware home contains files that are relevant only to the source node so you should remove the unnecessary files from the copy. You should exclude files in the `log`, `crs/init`, `racg/dump`, `srvm/log`, and `cdata` directories.

Use one of the following methods to exclude files from your backup file:

- Make a copy of the source CRS home and delete the unnecessary files from the copy.

  The following example shows the commands you can issue to remove unnecessary files from the copy of the CRS home. In the example, `crscluster` represents the name of the cluster:

  ```
  [root@node1 root]# cd /opt/oracle/product/11g/crs
  [root@node1 crs]# rm -rf ./opt/oracle/product/11g/crs/log/hostname
  [root@node1 crs]# find . -name '*.ouibak' -exec rm {} \;
  [root@node1 crs]# find . -name '*.ouibak.1' -exec rm {} \;
  ```

```
[root@node1 crs]# rm -rf ./cdata/*
[root@node1 crs]# rm -rf root.sh*
[root@node1 crs]# cd cfgtoollogs
[root@node1 cfgtoollogs]# find . -type f -exec rm -f {} \;
```

- Create an `excludeFileList` file and then use the `tar` command or Winzip to create a copy of the CRS home. For example, on Linux, issue the `tar cpfX - excludFileList.txt` command to create a tar file that does not excludes the unnecessary files.

Step 5 Create a copy of the source Oracle Clusterware home.

On the source node, create a copy of the Oracle Clusterware home using WinZip on Windows systems and tar or gzip on Linux and UNIX systems. Make sure that the tool that you use preserves the permissions and file timestamps.

When creating the copy, the best practice is to include the release number in the name of the file. For example, the following Linux example uses the `cd` command to change to the Oracle Clusterware home location, and then uses the `tar` command to create the copy named `crs11101.tgz`.

The following examples describe how to archive and compress the source Oracle Clusterware home on various platforms:

- On Linux and UNIX systems, issue the following command if you are using an `excludeFileList` file:

  ```
  tar cpfX - excludeFileList . | compress -fv > temp_dir/crs11101.tar.Z
  ```

  The following example shows the Linux and UNIX commands to create a copy when you are not using an `excludeFileList` file. In the `tar` command, the *pathname* variable represents the location of the file:

  ```
  [root@node1 root]# cd /opt/oracle/product/11g/crs/
  [root@node1 crs]# tar -zcvf /pathname/crs11101.tgz .
  ```

- On AIX or HPUX systems:

  ```
  tar cpf - . | compress -fv > temp_dir/crs11101.tar.Z
  ```

- On Windows systems, use WinZip to create a zip file

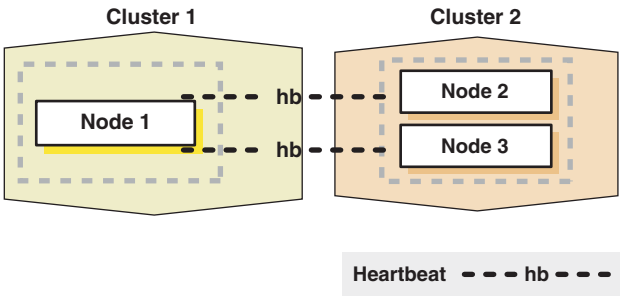
> **Note:** Do not use the jar utility to copy and compress the Oracle Clusterware home.

Cloning Oracle Clusterware to Create a New Cluster

This section explains how to create a new cluster by cloning a successfully installed Oracle Clusterware environment and copying it to the nodes on the destination cluster. The procedures in this section describe how to use cloning for Linux, UNIX, and Windows systems.

For example, you can use cloning to quickly duplicate a successfully installed Oracle Clusterware environment to create a new cluster. Figure 3–1 shows the end result of a cloning procedure in which the Oracle Clusterware home on Node 1 has been cloned to Node 2 and Node 3 on Cluster 2, making Cluster 2 a new two-node cluster.

Figure 3–1 Cloning to Create a New Oracle Clusterware Environment



At a high level, the steps to create a new cluster through cloning are as follows:

1. Prepare the new cluster nodes
2. Deploy Oracle Clusterware on the destination nodes
3. Run the clone.pl script on each destination node
4. Run the orainstRoot.sh script on each node
5. Run the CRS_home/root.sh script
6. Run the configuration assistants and the Oracle Cluster Verify utility

Step 1 Prepare the new cluster nodes

On each destination node, perform the following preinstallation steps:

- Specify the kernel parameters.
- Configure block devices for Oracle Clusterware devices.
- Ensure you have set the block device permissions correctly.
- Use short, nondomain-qualified names for all names in the Hosts file.
- Test whether or not the interconnect interfaces are reachable using the `ping` command.
- Verify that the VIP addresses are not active at the start of the cloning process by using the `ping` command (the `ping` command of the VIP address must fail).
- Run the Cluster Verification Utility (CVU) to verify your hardware and operating system environment.

See your platform-specific Oracle Clusterware installation guide for the complete preinstallation checklist.

> **Note:** Unlike traditional methods of installation, the cloning process does not validate your input during the preparation phase. (By comparison, during the traditional method of installation using the OUI, various checks take place during the interview phase.) Thus, if you make any mistakes during the hardware setup or in the preparation phase, then the cloned installation will fail.

Step 2 Deploy Oracle Clusterware on the destination nodes

Before you begin the cloning procedure described in this section, ensure that you have completed the prerequisite tasks to create a copy of the Oracle Clusterware home, as

described in the "Preparing the Oracle Clusterware Home for Cloning" section on page 3-3.

On each destination node, deploy the copy of the Oracle Clusterware home by performing the following steps:

1. If you do *not* have a shared Oracle Clusterware home, then restore the copy of the Oracle Clusterware home on each node in the destination cluster in the equivalent directory structure as the directory structure in which the Oracle Clusterware home resided on the source node. Skip this step if you have a shared Oracle Clusterware home.

   For example:

   - On Linux or UNIX systems, issue commands similar to the following:

     ```
     [root@node1 root]# mkdir -p /opt/oracle/product/11g/crs
     [root@node1 root]# cd /opt/oracle/product/11g/crs
     [root@node1 crs]# tar -zxvf /pathname/crs11101.tgz
     ```

     In the example, the *pathname* variable represents the directory structure in which you want to install the Oracle Clusterware home.

   - On Windows systems, unzip the Oracle Clusterware home on the destination node in the equivalent directory structure as the directory structure in which the Oracle Clusterware home resided on the source node.

2. Change the ownership of all files to `oracle:oinstall` group, and create a directory for the Oracle Inventory. For example, the following commands are for a Linux system:

   ```
   [root@node1 crs]# chown -R oracle:oinstall /opt/oracle/product/11g/crs
   [root@node1 crs]# mkdir -p /opt/oracle/oraInventory
   [root@node1 crs]# chown oracle:oinstall /opt/oracle/oraInventory
   ```

   Note: You can perform this step at the same time you perform steps 3 and 4 that run the `clone.pl` and `orainstRoot.sh` scripts on each cluster node.

3. Run the `preupgrade.sh` script from the `CRS_Home/install` directory on each target node as follows:

   ```
   @ preupgrade.sh -crshome target_crs_oh   -crsuser user_who_runs_cloning
   -noshutdown
   ```

   See Also: "Locating and Viewing Log Files Generated During Cloning" on page 3-14

Step 3 Run the clone.pl script on each destination node

To set up the new Oracle Clusterware environment, the `clone.pl` script requires you to provide a number of setup values to the script. You can supply the variables by either supplying input when you run the `clone.pl` script, or by creating a file in which you can assign values to the cloning variables. The following discussions describe these options.

- **Supplying Input to the `clone.pl` Script On the Command Line**

  If you do not have a shared Oracle Clusterware home, then on each destination node, then navigate to the `$ORACLE_HOME/clone/bin` directory and run the

clone.pl script, which performs the main Oracle Clusterware cloning tasks. To run the script, you need supply input to a number of variables, as shown in the following example:

```
perl clone.pl ORACLE_BASE=/opt/oracle ORACLE_HOME=CRS_home ORACLE_HOME_
NAME=Oracle_home_name '-On_storageTypeVDSK=2' '-On_storageTypeOCR=2'
'-O"sl_tableList={new_node:new_node-priv:new_node-vip}"' '-O-noConfig'
'-O"INVENTORY_LOCATION=central_inventory_location"'
```

Note: When cloning Oracle Clusterware using the clone.pl script, you must set a value for the ORACLE_BASE variable even though specifying Oracle Base is not a requirement of the Oracle Clusterware installation. You can set the ORACLE_BASE variable to any directory location (for example, you could set it to the CRS Home location), because the value is ignored.

The clone.pl script takes the following variables:

- *Oracle_home_name* is the name of the destination Oracle Clusterware home
- *new_node* is the name of the destination node
- *new_node-priv* is the private interconnect protocol address of the destination node
- *new_node-vip* is the virtual interconnect protocol address of the destination node
- *central_inventory_location* is the location of the Oracle central inventory

For example:

- On Linux and UNIX systems:

```
perl clone.pl ORACLE_BASE=/opt/oracle ORACLE_HOME=CRS_home ORACLE_HOME_
NAME=CRS_HOME_NAME '-On_storageTypeVDSK=2' '-On_storageTypeOCR=2'
'-O"sl_tableList={node2:node2-priv:node2-vip, node3:node3-priv:node3-vip}"'
'-O"ret_PrivIntrList=private interconnect list"'
'-O"sl_OHPartitionsAndSpace_valueFromDlg={partition and space
information}"'
 '-O-noConfig'
```

- On Windows systems:

```
perl clone.pl ORACLE_BASE=D:\oracle ORACLE_HOME=CRS_home ORACLE_HOME_
NAME=CRS_HOME_NAME '-On_storageTypeVDSK=2' '-On_storageTypeOCR=2'
'-O"sl_tableList={node2:node2-priv:node2-vip, node3:node3-priv:node3-vip}"'
 '-O"ret_PrivIntrList=private interconnect list"'
'-O"sl_OHPartitionsAndSpace_valueFromDlg={partition and space
information}"'
 '-O-noConfig' '-OPERFORM_PARTITION_TASKS=FALSE'
```

If you have a shared Oracle Clusterware home, then append the -cfs option to the command example in this step and provide a complete path location for the cluster file system. Ensure that the variables n_storageTypeOCR and n_storageTypeVDSK are set to 2 for redundant storage. Ensure that the values are set to 1 for nonredundant storage. In this case, you must also specify the mirror locations. On the other nodes, issue the same command, by passing an additional argument PERFORM_PARTITION_TASKS=FALSE.

For example:

```
perl clone.pl ORACLE_BASE=/opt/oracle ORACLE_HOME=CRS_home
ORACLE_HOME_NAME=CRS_home_name '-On_storageTypeVDSK=2'
'-On_storageTypeOCR=2' '-O"sl_tableList={node2:node2-priv:node2-vip,
 node3:node3-priv:node3-vip}"' '-O"ret_PrivIntrList=private interconnect list"'
'-O"sl_OHPartitionsAndSpace_valueFromDlg={partition and space information}"'
'-O-noConfig' '-OPERFORM_PARTITION_TASKS=FALSE'
```

> **See Also:** The "Cloning Script Variables Reference" section on page 3-11 for more information about setting these variables.

- **Supplying Input to the `clone.pl` Script in a File**

  Because the `clone.pl` script is sensitive to the parameters being passed to it, you must be accurate in your use of brackets, single quotation marks, and double quotation marks. To make running the clone.pl script less prone to errors, you can create a file that is similar to the `start.sh` script shown in Example 3–1 in which you can specify environment variables and cloning parameters to the `clone.pl` script.

  Example 3–1 shows an excerpt from the example file example called `start.sh` script that calls the `clone.pl` script and has been set up for a cluster named `crscluster`. Invoke the script as the operating system user that installed Oracle Clusterware.

Example 3–1 Excerpt From the start.sh Script to Clone Oracle Clusterware

```
#!/bin/sh
ORACLE_BASE=/opt/oracle
CRS_home=/opt/oracle/product/11g/crs
E01=CRS_home=/opt/oracle/product/11g/crs
E02=ORACLE_HOME=${CRS_home}
E03=ORACLE_HOME_NAME=OraCrs11g
E04=ORACLE_BASE=/opt/oracle
#C00="-O'-debug'"
C01="-O's_clustername=crscluster'"
C02="-O'INVENTORY_LOCATION=/opt/oracle/oraInventory'"
C03="-O'sl_tableList={node1:node1int:node1vip:N:Y,node2:node2int:node2vip:N:Y}'"
C04="-O'ret_PrivIntrList={eth0:144.25.212.0:1,eth1:10.10.10.0:2}'"
C05="-O'n_storageTypeVDSK=1'"
C06="-O's_votingdisklocation=/dev/sdc1' -O's_OcrVdskMirror1RetVal=/dev/sdd1'
            -O's_VdskMirror2RetVal=/dev/sde1'"
C07="-O'n_storageTypeOCR=1'"
C08="-O's_ocrpartitionlocation=/dev/sdc2' -O's_ocrMirrorLocation=/dev/sdd2'"

perl CRS_home/clone/bin/clone.pl $E01 $E02 $E03 $E04 $C01 $C02 $C03 $C04 $C05 $C06
$C07 $C08
```

The `start.sh` script sets several environment variables and cloning parameters, as described in Table 3–1 and Table 3–2, respectively.

Table 3–1 describes the environment variables E01, E02, E03, and E04 that are shown in bold typeface in Example 3–1.

Table 3–1 Environment Variables Passed to the clone.pl Script

Symbol	Variable	Description
E01	CRS_home	The location of the Oracle Clusterware home. This directory location must exist and must be owned by the Oracle operating system group: oinstall.
E02	ORACLE_HOME	The location of the Oracle Clusterware home. This directory location must exist and must be owned by the Oracle operating system group: oinstall.
E03	ORACLE_HOME_NAME	The name of the Oracle Clusterware home. This is stored in the Oracle Inventory.
E04	ORACLE_BASE	The location of the Oracle Base directory.

Also, see "Cloning Script Variables Reference" on page 3-11 for a description of the cloning parameters C01 through C08, that are shown in bold typeface in Example 3–1.

Step 4 Run the orainstRoot.sh script on each node

In the *Central Inventory* directory on each destination node, run the `orainstRoot.sh` script as the operating system user that installed Oracle Clusterware. This script populates the `/etc/oraInst.loc` directory with the location of the central inventory.

Note that you can run the script on each node simultaneously. For example:

```
[root@node1 root]# /opt/oracle/oraInventory/orainstRoot.sh
```

Ensure the `orainstRoot.sh` script has completed on each destination node before proceeding to the next step.

Step 5 Run the CRS_home/root.sh script

On each destination node, run the *`CRS_home`*`/root.sh` script. You can run the script on only one node at a time. The following example is for a Linux or UNIX system:

1. On the first node, issue the following command:

   ```
   [root@node1 root]# /opt/oracle/product/11g/crs/root.sh
   ```

   Ensure the *`CRS_home`*`/root.sh` script has completed on the first node before running it on the second node.

2. On each subsequent node, issue the following command:

   ```
   [root@node2 root]# /opt/oracle/product/11g/crs/root.sh
   ```

The `root.sh` script automatically sets up the node applications: Global Services Daemon (GSD), Oracle Notification Services (ONS), and Virtual IP (VIP) resources in the OCR.

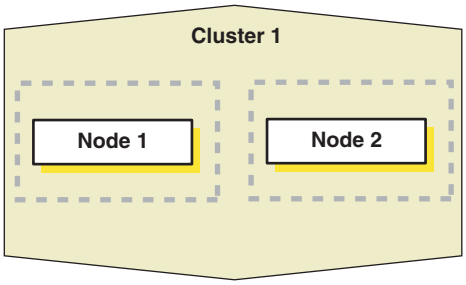
Step 6 Run the configuration assistants and the Oracle Cluster Verify utility

At the end of the Oracle Clusterware installation on each new node, run the configuration assistants and CVU using the commands in the `CRS_home/cfgtoollogs/configToolAllCommands` file.

Cloning to Extend Oracle Clusterware to More Nodes in the Same Cluster

For example, you can use cloning to quickly extend a successfully installed Oracle Clusterware environment to more nodes in the same cluster. Figure 3–2 shows the end result of a cloning procedure in which the Oracle Clusterware home on Node 1 has been cloned to Node 2 in the same cluster, making it a two-node cluster.

Figure 3–2 Cloning to Extend the Oracle Clusterware Environment to Another Node



At a high level, the steps to extend Oracle Clusterware to more nodes are nearly identical to the steps described in the "Cloning Oracle Clusterware to Create a New Cluster" section on page 3-4.

The following list describes the steps you perform to extend Oracle Clusterware to additional nodes in the cluster:

1. Prepare the new cluster nodes on page 3-5.
2. Deploy Oracle Clusterware on the destination nodes on page 3-5.
3. Run the `clone.pl` script on each destination node. The following example is for Linux or UNIX systems:

   ```
   perl clone.pl ORACLE_BASE=/opt/oracle ORACLE_HOME=$ORACLE_HOME ORACLE_HOME_
   NAME=Oracle_home_name "sl_tableList={node2:node2_priv:node2-vip}"
   INVENTORY_LOCATION=central_inventory_location -noConfig
   ```

4. Run the orainstRoot.sh script on each node on page 3-9 on each destination node.
5. Run the `addNode` script on the source node.

   Run the following command on the source node, where *new_node* is the name of the new node, *new_node-priv* is the private interconnect protocol address for the new node, and *new_node-vip* is the virtual interconnect protocol address for the new node:

   ```
   $ORACLE_HOME/oui/bin/addNode.sh -silent "CLUSTER_NEW_NODES=(new_nodes)"
    "CLUSTER_NEW_PRIVATE_NODE_NAMES=(new_node-priv)" "CLUSTER_NEW_VIRTUAL_
   HOSTNAMES=(new_node-vip)" -noCopy
   ```

   Note: Because the `clone.pl` script has already been run on the new node, this step only updates the inventories on the nodes and instantiates scripts on the local node

6. On the source node, run a script to instantiate the node:
   - On Linux and UNIX systems, run the `rootaddnode.sh` script from the *CRS_HOME*`/install` directory as `root` user.

- On Windows systems, run the `crssetup.add.bat` script from the `%`*CRS_HOME*`%\install` directory.

7. Run the CRS_home/root.sh script on page 3-9 on each destination node in Linux and UNIX environments.
8. Run the configuration assistants and the CLUVFY utility.

   As the user that owns the clusterware on the source node of the cluster, run the configuration assistants as described in the following steps:

   a. On Linux or UNIX systems, issue the following `onsconfig` command:

      ```
      onsconfig add_config node2:remote_port node3:remote_port
      ```

      You can obtain the remote port by issuing the `cat ons.config` command from the `/opmn/conf` directory in the CRS home location.

   b. On Windows systems, issue the racgons command:

      ```
      ./racgons add_config node2:remote_port node3:remote_port
      ```

   c. On Linux, UNIX, or Windows systems, run the CLUVFY utility in postinstallation verification mode to confirm that the installation of Oracle Clusterware was successful. For example:

      ```
      CRS_HOME/bin/cluvfy stage -post crsinst -n node1,node2
      ```

Cloning Script Variables Reference

Table 3–2 describes the variables that can be passed to the `clone.pl` script when you include the `-O` option on the command.

Table 3–2 Variables for the clone.pl Script with the -O option

Variable	Datatype	Description
`s_clustername`	String	Set the value for this variable to be the unique name of the cluster that you are creating from a cloning operation. Use a maximum of 15 characters. Valid characters for the cluster name can be any combination of lower and uppercase alphabetic characters A to Z, numerics 0 through 9 , hyphens (-), pound signs (#) and underscores (_).
`INVENTORY_LOCATION`	String	The location of the inventory. This directory location must exist and must be owned by the Oracle operating system group: `oinstall`.
`sl_tableList`	String List	A list of the nodes that make up the cluster. The format is a comma-delimited list of *`public_name`*:*`private_name`*:*`vip_name`*`:N:Y`. Set the value of this variable to be equal to the information in the cluster configuration information table. This file contains a comma-delimited list of values. The first field designates the public node name, the second field designates the private node name, and the third field designates the virtual host name. The fourth and fifth fields are used only by OUI and should default to `N:Y`. OUI parses these values and assign `s_publicname` and `s_privatename` variables accordingly. For example: `{"node1:node1-priv:node1-vip:N:Y","node2:node2-priv:node2-vip:N:Y"}`.

Table 3–2 (Cont.) Variables for the clone.pl Script with the -O option

Variable	Datatype	Description
ret_PrivIntrList	String List	This is the return value from the Private Interconnect Enforcement table. This variable has values in the format {Interface Name, Subnet, Interface Type}. The value for Interface Type can be one of the following: ■ 1 to denote public, ■ 2 to denote private ■ 3 to denote Do Not Use For example: {"eth0:10.87.24.0:2","eth1:140.87.24.0:1","eth3:140.74.30.0:3"} You can run the ipconfig command to identify the initial values from which you can determine the entries for ret_PrivIntrList.
n_storageTypeVDSK	Integer	If you are using: ■ A single voting disk, set this parameter to 1 (not redundant). ■ Multiple voting disks, set this parameter to 2 (redundant).
n_storageTypeOCR	Integer	If you are using: ■ A single OCR disk, set this parameter to 1 (not redundant). ■ Multiple OCR disks, set this parameter to 2 (redundant).
s_clustername	String	This variable contains user-entered cluster name information; allow a maximum of 15 characters.
VdskMirrorNotReqd	String	This variable is not required in the Oracle Cluster Registry (OCR) dialog.
CLUSTER_CONFIGURATION_FILE	String	This variable is used to pass the cluster configuration file information which is the same file as that specified during installation. You may use this file instead of sl_tablelist. This file contains the public node name, private node name, and virtual host name which is white space-delimited information for the nodes of the cluster. For example, node1 node1-priv node1-vip node2 node2-priv node2-vip Note that if you are cloning from an existing installation, then you should use sl_tableList. Do not specify this variable for a clone installation.
s_votingdisklocation	String	Set the value of this variable to be the location of the voting disk. For example: /oradbshare/oradata/vdisk If you are using: ■ A single voting disk, only specify the voting disk location with the s_votingdisklocation parameter. ■ Multiple voting disks, set the s_votingdisklocation, s_OcrVdskMirror1RetVal, and the s_VdskMirror2RetVal parameters.

Table 3–2 (Cont.) Variables for the clone.pl Script with the -O option

Variable	Datatype	Description
s_OcrVdskMirror1RetVal	String	Set the value of this variable to be the location of the first additional voting disk. You must set this variable if you choose a value of 1 for the n_storageTypeVDSK variable or Not Redundant. For example: /oradbshare/oradata/vdiskmirror1
s_ocrpartitionlocation	String	Set the value of this variable to the OCR location. Oracle places this value in the ocr.loc file when you run the root.sh script. For example: /oradbshare/oradata/ocr If you are using: ■ A single OCR disk, only set the s_ocrpartitionlocation parameter to specify the location of the OCR partition. ■ Multiple OCR disks, set the s_ocrpartitionlocation parameter and the s_ocrMirrorLocation parameter.
s_ocrMirrorLocation	String	Set the value of this variable to the value for the OCR mirror location. Oracle places this value in the ocr.loc file when you run the root.sh script. You must set this variable if you choose a value of 1 for the n_storageTypeOCR variable or Not Redundant. For example: /oradbshare/oradata/ocrmirror
s_VdskMirror2RetVal	String	Set the value of this variable to be the location of the second additional voting disk. You must set this variable if you choose a value of 1 for the n_storageTypeVDSK variable or Not Redundant. /oradbshare/oradata/vdiskmirror2
CLUSTER_NODES	String List	The value of this variable represents the cluster node names that you selected for installation. For example, if you selected node1: CLUSTER_NODES = {"node1"}
b_Response	Boolean	Only set this variable when performing a silent installation with a response file. The valid values are true or false.

Table 3–2 (Cont.) Variables for the clone.pl Script with the -O option

Variable	Datatype	Description
`sl_OHPartitionsAndSpace_valueFromDlg`	String List	Set the value for this variable using the following format: `1` = disk number `2` = partition number `3` = partition size `4` = format type, `0` for raw and `1` for cluster file system `5` = Drive Letter (this value is not applicable if you use raw devices, use the available drive letter if you are using a cluster file system. `6` Usage type values: ■ `0` = Data or software use *only* ■ `1` = Primary OCR *only* ■ `2` = Voting disk *only* ■ `3` = Primary OCR and voting disk on the same cluster file system partition ■ `4` = OCR mirror *only* ■ `5` = OCR mirror and voting disk on the same cluster file system partition For example, to configure the OCR and voting disk on raw devices and to not use a cluster file system for either data or software, set `sl_OHPartitionsAndSpace_valueFromDlg` to list only the partitions that you intend to use for an Oracle Clusterware installation using the following format: `sl_OhPartitionsAndSpace_valueFromDlg = {Disk,Partition,partition size, 0,N/A,1,Disk,Partition, partition size,0,N/A,2,.....)`

Locating and Viewing Log Files Generated During Cloning

The cloning script runs multiple tools, each of which may generate its own log files. After the `clone.pl` script finishes running, you can view log files to obtain more information about the cloning process.

The following log files that are generated during cloning are the key log files of interest for diagnostic purposes:

- *Central_Inventory*`/logs/cloneActions timestamp.log`

  Contains a detailed log of the actions that occur during the OUI part of the cloning.

- *Central_Inventory*`/logs/oraInstall timestamp.err`

  Contains information about errors that occur when OUI is running.

- *Central_Inventory/logs/oraInstall timestamp.out*

  Contains other miscellaneous messages generated by OUI.

- `$ORACLE_HOME/clone/logs/clone timestamp.log`

  Contains a detailed log of the actions that occur prior to cloning as well as during the cloning operations.

- `$ORACLE_HOME/clone/logs/error timestamp.log`

Contains information about errors that occur prior to cloning as well as during cloning operations.

Table 3–3 describes how to find the location of the Oracle inventory directory.

Table 3–3 Finding the Location of the Oracle Inventory Directory

Type of System ,,,	Location of the Oracle Inventory Directory
All UNIX computers except Linux and IBM AIX	/var/opt/oracle/oraInst.loc
IBM AIX and Linux	`/etc/oraInst.loc` file.
Windows	Obtain the location from the Windows Registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\INST_LOC`

4

Adding and Deleting Oracle Clusterware Homes

This chapter describes how to use the `addNode.sh` and `rootdeletenode.sh` scripts to copy the Oracle Clusterware home from an existing Oracle Clusterware home to other nodes. This chapter provides instructions for Linux and UNIX systems.

You should use the add node procedures described in this chapter to add or delete Oracle Clusterware from nodes in the cluster. If your goal is create new clusters or extend Oracle Clusterware to more nodes in the same cluster, then use the cloning procedures that are described in Chapter 3.

The topics in this chapter include the following:

- Prerequisite Steps for Adding Oracle Clusterware
- Adding and Deleting Oracle Clusterware Homes on Linux and UNIX Systems

Prerequisite Steps for Adding Oracle Clusterware

The following steps assume that you already have an operative Linux or UNIX environment.

Complete the following steps to prepare the new nodes in the cluster:

1. Make physical connections

   Connect the new nodes' hardware to the network infrastructure of your cluster. This includes establishing electrical connections, configuring network interconnects, configuring shared disk subsystem connections, and so on. See your hardware vendor documentation for details about this step.

2. Install the operating system

   Install a cloned image of the operating system that matches the operating system on the other nodes in your cluster. This includes installing required service patches and drivers. See your hardware vendor documentation for details about this process.

3. Create Oracle users.

   As `root` user, create the Oracle users and groups using the same user ID and group ID as on the existing nodes.

4. Verify the installation with the Cluster Verification Utility (CVU) using the following steps:

   a. From the `/bin` directory in the *`CRS_home`* on the existing nodes, run the CVU command to verify your installation at the post hardware installation stage as

shown in the following example, where *node_list* is a comma-delimited list of nodes you want in your cluster:

```
cluvfy stage -post hwos -n node_list|all [-verbose]
```

This command causes CVU to verify your hardware and operating system environment at the post-hardware setup stage. After you have configured the hardware and operating systems on the new nodes, you can use this command to verify the node is reachable, for example, to all of the nodes from the local node. You can also use this command to verify user equivalence to all given nodes the local node, node connectivity among all of the given nodes, accessibility to shared storage from all of the given nodes, and so on.

Note: You can only use the `all` option with the `-n` argument if you have set the `CV_NODELIST` variable to represent the list of nodes on which you want to perform the CVU operation.

See Also: Appendix A, "Cluster Verification Utility Reference" for more information

b. From the `/bin` directory in the *CRS_home* on the existing nodes, run the CVU command to obtain a detailed comparison of the properties of the reference node with all of the other nodes that are part of your current cluster environment where *ref_node* is a node in your existing cluster against which you want CVU to compare, for example, the newly added nodes that you specify with the comma-delimited list in *node_list* for the `-n` option, *orainventory_group* is the name of the Oracle inventory group, and *osdba_group* is the name of the OSDBA group:

```
cluvfy comp peer [ -refnode ref_node ] -n node_list
[ -orainv orainventory_group ] [ -osdba osdba_group ] [-verbose]
```

Note: For the reference node, select a node from your existing cluster nodes against which you want CVU to compare, for example, the newly added nodes that you specify with the `-n` option.

---

5. Check the installation

To verify that your installation is configured correctly, perform the following steps:

a. Ensure that the new nodes can access the private interconnect. This interconnect must be properly configured before you can complete the procedures described in this chapter.

b. If you are not using a cluster file system, then determine the location on which your cluster software was installed on the existing nodes. Make sure that you have at least 250MB of free space on the same location on each of the new nodes to install Oracle Clusterware. In addition, ensure you have enough free space on each new node to install the Oracle binaries.

c. Ensure that the Oracle Cluster Registry (OCR) and the voting disk are accessible by the new nodes using the same path as the other nodes use. In addition, the OCR and voting disk devices must have the same permissions as on the existing nodes.

d. Verify user equivalence to and from an existing node to the new nodes using rsh or ssh on Linux and UNIX systems, or on Window systems make sure that you can run the following command from *all* of the existing nodes of your cluster where the `hostname` is the public network name of the new node:

```
NET USE \\hostname\C$
```

You have the required administrative privileges on each node if the operating system responds with:

```
Command completed successfully.
```

After completing the procedures in this section, your new nodes are connected to the cluster and configured with the required software to make them visible to Oracle Clusterware.

Note: Avoid changing host names after you complete the Oracle Clusterware installation, including adding or deleting domain qualifications. Nodes with changed host names must be deleted from the cluster and added back with the new name.

Adding and Deleting Oracle Clusterware Homes on Linux and UNIX Systems

This section explains Oracle Clusterware home addition and deletion on Linux and UNIX systems and it assumes that you have already performed the steps in the "Prerequisite Steps for Adding Oracle Clusterware" section.

For node addition, ensure that you install the required operating system patches and updates on the new nodes. Then configure the new nodes to be part of your cluster at the network level. Use the instructions in this section to extend the Oracle Clusterware home from an existing Oracle Clusterware home to the new nodes

Finally, you can optionally extend the Oracle database software with Oracle RAC components to the new nodes and make the new nodes members of the existing Oracle RAC database. See the node addition procedures described in *Oracle Real Application Clusters Administration and Deployment Guide*.

This section includes the following topics:

- Adding an Oracle Clusterware Home to a New Node On Linux or UNIX Systems
- Deleting an Oracle Clusterware Home from a Linux or UNIX System

Adding an Oracle Clusterware Home to a New Node On Linux or UNIX Systems

This section describes how to use Oracle Universal Installer (OUI) to add an Oracle Clusterware home to a node in your cluster. This documentation assumes:

- There is an existing cluster that has a node named `node1`
- You are adding Oracle Clusterware from `node2`
- You have already successfully installed Oracle Clusterware on `node1` in a nonshared home, where *`CRS_home`* represents the successfully installed home

You can use either of the following procedures to add an Oracle Clusterware home to a node:

Adding an Oracle Clusterware Home to a New Node Using OUI in Interactive Mode

This procedure assumes that you have performed the tasks outlined in "Prerequisite Steps for Adding Oracle Clusterware" on page 4-1. OUI requires access to the private interconnect that you verified as part of the installation validation in Step 1. If OUI cannot make the required connections, then you will not be able to complete the following steps to add Oracle Clusterware to other nodes.

Note: Instead of performing the first six steps of this procedure, you can alternatively run the `addNode.sh` script in silent mode as described "Adding an Oracle Clusterware Home to a New Node Using OUI in Silent Mode" on page 4-5.

---

1. Ensure that you have successfully installed Oracle Clusterware on at least one node in your cluster environment. Also, for these procedures to complete successfully, you must ensure that *`CRS_home`* identifies your successfully installed Oracle Clusterware home.
2. Start OUI:

   Go to *`CRS_home`*`/oui/bin` and run the `addNode.sh` script on one of the *existing* nodes. The OUI runs in `add node` mode and the OUI Welcome page appears. Click **Next** and the Specify Cluster Nodes for Node Addition page displays.
3. OUI displays the Node Selection Page on which you should select the node or nodes that you want to add and click **Next**.

   The upper table on the Specify Cluster Nodes for Node Addition page shows the existing nodes, the private node names, and the virtual IP (VIP) addresses that are associated with Oracle Clusterware. Use the lower table to enter the public, private node names and the virtual hostnames of the new nodes.
4. Verify the entries that OUI displays on the Summary Page and click **Next**.

   If any verifications fail, then OUI redisplays the Specify Cluster Nodes for Node Addition page with a Status column in both tables indicating errors. Correct the errors or deselect the nodes that have errors and proceed. However, you cannot deselect existing nodes; you must correct problems on nodes that are already part of your cluster before you can proceed with node addition. If all the checks succeed, then OUI displays the Node Addition Summary page.
5. The Node Addition Summary page displays the following information showing the products that are installed in the Oracle Clusterware home that you are extending to the new nodes:
   - The source for the add node process, which in this case is the Oracle Clusterware home
   - The private node names that you entered for the new nodes
   - The new nodes that you entered
   - The required and available space on the new nodes
   - The installed products listing the products that are already installed on the existing Oracle Clusterware home

Click **Next** and OUI displays the Cluster Node Addition Progress page.

6. The Cluster Node Addition Progress page shows the status of the cluster node addition process. The table on this page has two columns showing the four phases of the node addition process and the phases' statuses as follows:

   - Instantiate Root Scripts—Instantiates `rootaddNode.sh` with the public nodes, private node names, and virtual hostnames that you entered on the Cluster Node Addition page.
   - Copy the Oracle Clusterware home to the New Nodes—Copies the Oracle Clusterware home to the new nodes unless the Oracle Clusterware home is on a cluster file system.
   - Save Cluster Inventory—Updates the node list associated with the Oracle Clusterware home and its inventory.
   - Run `rootaddNode.sh` and `root.sh`—Displays a dialog prompting you to run the `rootaddNode.sh` script[1] from the local node (the node on which you are running OUI) and to run the `root.sh` script[2] on the new nodes. If OUI detects that the new nodes do not have an inventory location, then OUI instructs you to run the `orainstRoot.sh` script[3] on those nodes. The central inventory location is the same as that of the local node. The `addNodeActions`*`timestamp`*`.log` file, where *`timestamp`* shows the session start date and time, contains information about which scripts you need to run and on which nodes you need to run them.

   The Cluster Node Addition Progress page's Status column displays `In Progress` while the phase is in progress, `Suspended` when the phase is pending execution, and `Succeeded` after the phase completes. On completion, click **Exit** to end the OUI session. After OUI displays the End of Node Addition page, click **Exit** to end the OUI session.

7. Run the configuration assistants and CVU using the commands in the `CRS_home/cfgtoollogs/configToolAllCommands` file, replacing the existing node name with the new node name for each command.

Adding an Oracle Clusterware Home to a New Node Using OUI in Silent Mode

1. Ensure that you have successfully installed Oracle Clusterware on at least one node in your cluster environment. To perform the following procedure, the *`CRS_home`* must identify your successfully installed Oracle Clusterware home.
2. Go to *`CRS_home`*`/oui/bin` and run the `addNode.sh` script using the following syntax where `node2` is the name of the new node that you are adding, `node2-priv` is the private node name for the new node, and `node2-vip` is the VIP name for the new node:

   ```
   ./addNode.sh -silent "CLUSTER_NEW_NODES={node2}"
   "CLUSTER_NEW_PRIVATE_NODE_NAMES={node2-priv}"
   "CLUSTER_NEW_VIRTUAL_HOSTNAMES={node2-vip}"
   ```

   Alternatively, you can specify the `variable=value` entries in a response file and run the `addNode` script as follows:

---

[1] Run the `rootaddNode.sh` script from the `CRS_home/install/` directory on the node from which you are running OUI.

[2] Run the root.sh script on the new node from the Oracle Clusterware home to start Oracle Clusterware on the new node.

[3] Run the `orainstRoot.sh` script on the new node if OUI prompts you to do so.

```
addNode.sh  -responseFile filename OR addNode.bat  -responseFile filename
```

See Also: *Oracle Universal Installer and OPatch User's Guide* for details about how to configure command-line response files

Note: command-line values always override response file values.

3. Perform step 7 in the "Adding an Oracle Clusterware Home to a New Node Using OUI in Interactive Mode" section.

Deleting an Oracle Clusterware Home from a Linux or UNIX System

The procedures for deleting an Oracle Clusterware home assume that you have successfully installed the Oracle Clusterware on the node from which you want to delete the Oracle Clusterware home. You can use either of the following procedures to delete an Oracle Clusterware home from a node:

- Deleting an Oracle Clusterware Home Using OUI in Interactive Mode
- Deleting an Oracle Clusterware Home Using OUI in Silent Mode

Note: Oracle recommends that you back up your voting disk and OCR files after you complete the node deletion process.

Deleting an Oracle Clusterware Home Using OUI in Interactive Mode

Use the following steps to remove Oracle Clusterware from a cluster node.

Step 1 Verify the location of the Oracle Clusterware home

Ensure that *CRS_home* correctly specifies the full directory path for the Oracle Clusterware home on each node, where *CRS_home* is the location of the installed Oracle Clusterware software

Step 2 Remove the stored network configuration

If you ran the Oracle Interface Configuration Tool (OIFCFG) with the `-global` option during the installation, then skip this step.

Otherwise, from a node that is going to remain in your cluster, in the *CRS_home*`/bin` directory, run the following command where `node2` is the name of the node that you are deleting:

```
./oifcfg delif -node node2
```

Step 3 Obtain the remote port number

Obtain the remote port number, which you will use in the next step. To do this, issue the following command from the *CRS_home*`/opmn/conf` directory:

```
cat ons.config
```

Step 4 Remove the ONS daemon configuration

From *CRS_home*`/bin` on a node that is going to remain in the cluster, run the Oracle Notification Service Utility (RACGONS). In the following example, the *remote_port* variable represents the ONS remote port number that you obtained in step 3 and `node2` is the name of the node that you are deleting:

```
./racgons remove_config node2:remote_port
```

Step 5 Disable the Oracle Clusterware applications

On the node to be deleted, run the `rootdelete.sh` script as the `root` user from the *`CRS_home/install`* directory to disable the Oracle Clusterware applications and daemons running on the node. If you are deleting Oracle Clusterware from more than one node, then perform this step on each node that you are deleting.

Step 6 Delete the node and update the cluster registry

From any node that you *are not deleting*, issue the following command from the *`CRS_home/install`* directory as the `root` user to delete the node from the cluster and to update the Oracle Cluster Registry (OCR). In the following command, the variable *`node2,node2-number`* represents the node and the node number that you want to delete:

```
./rootdeletenode.sh node2,node2-number
```

If necessary, identify the node number using the following command on the node that you are deleting:

```
CRS_home/bin/olsnodes -n
```

Step 7 Remove the node from the node list

On the node that is to be deleted, run the following command from the *`CRS_home`*`/oui/bin` directory where *`node_to_be_deleted`* is the name of the node that you are deleting:

```
./runInstaller -updateNodeList ORACLE_HOME=CRS_home
"CLUSTER_NODES={node_to_be_deleted}"
CRS=TRUE -local
```

Step 8 Detach or deinstall the Oracle Clusterware software

On the node that you are deleting, run OUI using the `runInstaller` command from the *`CRS_home`*`/oui/bin` directory. Depending on whether you have a shared or nonshared Oracle home, complete one of the following procedures:

- If you have a shared home, then on any node other than the node to be deleted, run the following command from the *`CRS_home`*`/oui/bin` directory:

  ```
  ./runInstaller -detachHome  ORACLE_HOME=CRS_home
  ```

- For a nonshared home, deinstall the Oracle Clusterware home from the node that you are deleting using OUI as follows by issuing the following command from the *`Oracle_home`*`/oui/bin` directory, where *`CRS_home`* is the name defined for the Oracle Clusterware home:

  ```
  ./runInstaller -deinstall "REMOVE_HOMES={CRS_home}"
  ```

Step 9 Update the node list on the remaining nodes

On any node other than the node you are deleting, run the following command from the *`CRS_home`*`/oui/bin` directory where *`remaining_nodes_list`* is a comma-delimited list of the nodes that are going to remain part of your cluster:

```
./runInstaller -updateNodeList ORACLE_HOME=CRS_home
"CLUSTER_NODES={remaining_nodes_list}"
CRS=TRUE
```

Deleting an Oracle Clusterware Home Using OUI in Silent Mode

Use the following steps to remove Oracle Clusterware from a cluster node.

1. Ensure that *CRS_home* correctly identifies the Oracle Clusterware home on each node.
2. Detach or deinstall Oracle Clusterware.

   Depending on whether you have a shared or nonshared Oracle Clusterware home, complete one of the following two procedures:

   - For shared homes, do not perform a deinstallation operation. Instead, perform a detach home operation on the node that you are deleting. To do this, run the following command from *CRS_home*/oui/bin:

     ```
     ./runInstaller -detachHome ORACLE_HOME=CRS_home
     ```

   - For a nonshared home, deinstall the Oracle Clusterware home from the node that you are deleting using OUI as follows by issuing the following command from the *Oracle_home*/oui/bin directory, where *CRS_home* is the name defined for the Oracle Clusterware home:

     ```
     ./runInstaller -deinstall -silent "REMOVE_HOMES={CRS_home}"
     ```

3. Update the node list.

   On any node other than the node you are deleting, run the following command from the *CRS_home*/oui/bin directory where *remaining_nodes_list* is a comma-delimited list of the nodes that are going to remain part of your cluster:

   ```
   ./runInstaller -updateNodeList ORACLE_HOME=CRS_home
   "CLUSTER_NODES={remaining_nodes_list}"
   CRS=TRUE
   ```

5

Making Applications Highly Available Using Oracle Clusterware

This chapter explains how you can extend the high availability of the Oracle Clusterware framework to your applications. You do this by wrapping your applications with Oracle Clusterware commands. That is, you can use the same high availability mechanisms of the Oracle database and Oracle Real Application Clusters (Oracle RAC) to make your custom applications highly available. You can use Oracle Clusterware to monitor, relocate, and restart your applications as described in this chapter under the following topics:

- Overview of Managing Custom Applications with Oracle Clusterware Commands
- Creating Application Profiles
- Example of Using Oracle Clusterware Commands to Create Application Resources
- Oracle Clusterware Action Program Guidelines
- Using Oracle Clusterware Commands

> **See Also:** Appendix D, "High Availability Oracle Clusterware Command-Line Reference and C API" for detailed information about Oracle Clusterware commands

Overview of Using Oracle Clusterware Commands to Enable High Availability

Oracle Clusterware includes a high availability framework that provides an infrastructure to protect any application. Oracle Clusterware ensures that applications that it manages start when the system starts. Oracle Clusterware also monitors the applications to make sure that they are always available. For example, if a process fails, then Oracle Clusterware attempts to restart the process based on scripts that you customize. If a node in the cluster fails, then you can program processes that normally run on the failed node to restart on another node. The monitoring frequency, starting, and stopping of the applications and the application dependencies are configurable.

To make applications highly available, first create an application profile that identifies your application. The application profile uses a second component, an action program, that describes how Oracle Clusterware should monitor your application and how Oracle Clusterware should respond to changes in your application's status. Oracle stores application profile attributes in the OCR. The definitions of an application profile, action program, and the other primary Oracle Clusterware high availability components are as follows:

- **Action Program**—A program that defines the location of your program, the monitoring frequency, and the start and stop actions that Oracle Clusterware should perform on the application. The start action starts the application, the stop action stops the application, and the monitoring or check action checks the application's status.
- **Application Profile**—An Oracle Clusterware resource file that describes the attributes of your application. An application profile influences your application's behavior and it identifies other programs or scripts that Oracle Clusterware should run to perform various actions.
- **Privileges**—Access and usage privileges that enable Oracle Clusterware to control all of the components of your application for high availability operations, including the right to start processes under other user identities. Oracle Clusterware must run as a privileged user to control applications with the correct start and stop processes. On Linux and UNIX platforms, this usually implies that Oracle Clusterware must run as the `root` user and on Windows platforms Oracle Clusterware must run as `Administrator`.
- **Resource**—An entity that Oracle Clusterware manages for high availability such as your application.

  Note: A resource in the Oracle Clusterware context is not the same as a resource in the Oracle database sense, such as "Resource Manager." A resource that Oracle Clusterware refers to any entity managed by Oracle Clusterware, including application programs.

- **Resource Dependency**—A relationship among resources or applications that implies an operational ordering. For example, during a start operation, parent resources are started before resources that have dependencies. Stopping a resource is prevented if resources that depend on it are running,. However, you can force the termination with the `crs_stop -f` command in which case all resources that depend on the resource being stopped are stopped first.
- **Oracle Cluster Registry (OCR)**—A mechanism that stores configuration information that Oracle Clusterware and other Oracle RAC manageability systems use. The OCR uses a hierarchical name space for key value pairs. Keys and subkeys have enforced `user`, `group`, and `other` permissions.
- **Template**—A text file generated by the `crs_profile` command that contains the default values for application profile attributes.

  See Also: Appendix D, "High Availability Oracle Clusterware Command-Line Reference and C API" for more information about using Oracle Clusterware commands to make your applications highly available

Overview of Managing Custom Applications with Oracle Clusterware Commands

You can use Oracle Clusterware commands to start, stop, relocate, and check the status of your custom applications. Do this by defining your application with an application profile. The profile defines attributes that affect how Oracle Clusterware manages your application. Then register your application information in the OCR using the `crs_register` command. Use the following steps to create an application profile:

1. Create an application profile by running the `crs_profile` command. See Table 5–1 for a listing of required and optional entries for the Oracle Clusterware profiles.
2. Register the application profile using the `crs_register` command.
3. Run the `crs_start` command to initiate the application profile and then Oracle Clusterware runs the `start` command that you have included in the profile to start your application.
4. Oracle Clusterware periodically runs the `action program` command to check an application's status.
5. In the event of a check or node failure, Oracle Clusterware recovers the application either by restarting it on the current node or by relocating the application to another node.
6. If you run the `crs_stop` command to stop the application, then Oracle Clusterware runs the `stop action program` command to stop it.

You can manage application availability as follows:

- Specify starting resources during cluster or node start up
- Restart applications that fail
- Relocate applications to other nodes if they cannot run in their current location

Full administrative privileges are not required when using Oracle Clusterware. Any user can create resources or applications. However, the creator or owner must grant permission to other users or user groups in order for others to be able to use Oracle Clusterware on those applications. Additionally, profiles that have privileges defined can only be modified by privileged users. The following sections provide further details about application profiles.

Note: Do not use Oracle Clusterware commands prefixed with `crs_` (except for `crs_stat`) on resources that have names beginning with the prefix `ora` unless Oracle Support Services asks you to. Instead, use the Server Control (SRVCTL) utility on Oracle resources. You can create resources that depend on resources that Oracle has defined. When creating resources, do not use an `ora` prefix in the resource name. This prefix is reserved for Oracle use only.

Creating Application Profiles

Application profiles have attributes that define how Oracle Clusterware starts, manages, and monitors applications. One attribute is the location of the `action program` that Oracle Clusterware uses to manipulate the application. Oracle Clusterware uses the `action program` to monitor or check the application status and to start and stop it. Oracle reads application profiles from files stored in specified locations and stores the information in the OCR. You use Oracle Clusterware commands in profiles to designate resource dependencies and to determine what happens to an application or service when it loses access to a resource on which it depends.

The following section describes profiles in more detail. The recommended method for creating profiles is to use the `crs_profile` command, which is described in detail in Appendix D, "High Availability Oracle Clusterware Command-Line Reference and C API" on page D-4.

Application Resource Profiles

Attributes are defined by `name=value` entries in profile files and these entries can be in any order in the file. The following are some of the primary attributes of an application profile:

- Resources that are required by an application which are defined by settings for the `REQUIRED_RESOURCES` parameter. Oracle Clusterware relocates or stops an application if a required resource becomes unavailable. Required resources are defined for each node.
- Rules for choosing the node on which to start or restart an application are defined by settings for the `PLACEMENT` parameter. The application must be accessible by the nodes that you nominate for placement.
- A list of nodes to use in order of preference when Oracle Clusterware starts or fails over an application which is defined by settings for the `HOSTING_MEMBERS` parameter. This list is used if the placement policy defined by `the PLACEMENT` parameter is favored or restricted.
- The filenames of application profiles must be in the form *`resource_name`*`.cap` where *`resource_name`* is the name that you or the system assigns to an application and `cap` is the file suffix. The Oracle Clusterware commands in profiles refer to applications by name, such as `resource_name`, but not by the full filename.

Required and Optional Profile Attributes

Application profiles have optional and required profile attributes. Optional profile attributes may be left unspecified in the profile. Optional profile attributes that have default values are merged at registration time with the values that are stored in the template for that resource type and for the generic template. Default values are derived from the template.

Each resource type has a template file named `TYPE_resource_type.cap` that is stored in the `template` subdirectory under the `crs` directory of the Oracle Clusterware home. A generic template file for values that are used in all types of resources is stored in the same location in the file named `TYPE_generic.cap`.

Application Profile Attributes

Table 5–1 lists the Oracle Clusterware application profile attributes in alphabetical order. For each attribute, the table shows whether the attribute is required, its default value, and an attribute description.

Table 5–1 Application Profile Attributes

Attribute	Required	Default	Description
ACTION_ SCRIPT	Yes	None	The resource-specific script for starting, stopping, and checking a resource. You may specify a full path for the action program file. Otherwise, the default paths are used: *CRS_home*/crs/script for privileged, and *CRS_home*/crs/public for public. You may also specify a relative path with this default path as the starting point.
ACTIVE_ PLACEMENT	No	0	When set to 1, Oracle Clusterware reevaluates the placement of a resource during addition or restart of a cluster node.
AUTO_START	No	always	Indicates whether Oracle Clusterware should automatically start a resource after a cluster restart. Valid AUTO_START values are: ■ always—Causes the resource to restart when the node restarts regardless of the resource's state when the node stopped. ■ restore—Does not start the resource at restart time if it was in an offline state, such as STATE=OFFLINE, TARGET=OFFLINE, when the node stopped. The resource is restored to its state when the node went down. The resource is started only if it was online before and not otherwise. ■ never—Oracle Clusterware never restarts the resource regardless of the resource's state when the node stopped. **Note:** Oracle only supports lower-case values for always, restore, and never.
CHECK_ INTERVAL	No	60	The time interval, in seconds, between repeated executions of the check entry point of a resource's action program. There can be some overhead associated if you set the check interval to a low value and enable frequent checks.
DESCRIPTION	No	Name of the resource	A description of the resource.
FAILOVER_ DELAY	No	0	The amount of time, in seconds, that Oracle Clusterware waits before attempting to restart or fail over a resource.
FAILURE_ INTERVAL	No	0	The interval, in seconds, during which Oracle Clusterware applies the failure threshold. If the value is zero (0), then tracking of failures is disabled.
FAILURE_ THRESHOLD	No	0	The number of failures detected within a specified FAILURE_ INTERVAL before Oracle Clusterware marks the resource as unavailable and no longer monitors it. If a resource's check script fails this number of times, then the resource is stopped and set offline. If the value is zero (0), then tracking of failures is disabled. The maximum value is 20.

Table 5–1 (Cont.) Application Profile Attributes

Attribute	Required	Default	Description
HOSTING_ MEMBERS	Sometimes	None	An ordered list of cluster nodes separated by blank spaces that can host the resource. This attribute is required only if PLACEMENT equals favored or restricted. This attribute must be empty if PLACEMENT equals balanced. Enter node names as values for the HOSTING_MEMBERS attribute, not virtual host names or physical host names. Use the node names that you used when you installed Oracle Clusterware. The resources that you mention should contain the node name for the node on which they run. Run the olsnodes commands to see your node names. The HOSTING_MEMBERS Oracle Clusterware attribute is set automatically when these Oracle Clusterware resources are created; you do not need to take further action to ensure that the attribute is set. The node name is usually the same as the physical host name. However, it can be different. For example, when vendor clusterware is present, the Oracle Clusterware nodes are named the same as the vendor clusterware nodes. Not all vendor clusterware implementations use the physical node names as node names. Use the lsnodes command to display vendor clusterware node names. When there is no vendor clusterware, then the Oracle Clusterware node names must be the same as the physical hostname.
NAME	Yes	None	The name of the application. The application name is a string that contains a combination of letters a-z or A-Z, and digits 0-9. The naming convention is to start with an alphanumeric prefix, such as sky1, and complete the name with an identifier to describe it. The name can contain any platform-supported characters except the exclamation point (!). However, the application name cannot begin with a period.
OPTIONAL_ RESOURCES	No	None	An ordered list of resource names separated by blank spaces that this resource uses during placement decisions. Up to 58 user-defined resources can be listed.
PLACEMENT	No	balanced	The placement policy (balanced, favored, or restricted) specifies how Oracle Clusterware chooses the cluster node on which to start the resource. Also, see "Application Placement Policies" on page 5-11.
REQUIRED_ RESOURCES	No	None	An ordered list of resource names separated by blank spaces that this resource depends on. Each resource to be used as a required resource in this profile must be registered with Oracle Clusterware or the resource's profile registration will fail.
RESTART_ ATTEMPTS	No	1	The number of times that Oracle Clusterware attempts to restart a resource on a single cluster node before attempting to relocate the resource. A value of 1 means that Oracle Clusterware only attempts to restart the resource once on a node. A second failure causes an attempt to relocate the resource.
RESTART_ COUNT			The counter maintained by the Oracle Clusterware daemon for the number of times that a resource had been restarted. It goes from zero to RESTART_ATTEMPTS. This is also written to the OCR.
SCRIPT_ TIMEOUT	No	60	The maximum time (in seconds) for an action script to execute. An error message is returned if the script does not complete within the time specified. The timeout applies to all action script entry points (start, stop, and check). If you do not specify a value, Oracle Clusterware assumes a default value of 60 seconds.
START_ TIMEOUT			The maximum time (in seconds) in which a start action script can run. An error message is returned if the script does not complete within the time specified. If you do not specify this attribute or you specify 0 seconds, then Oracle Clusterware uses the SCRIPT_TIMEOUT value.

Table 5–1 (Cont.) Application Profile Attributes

Attribute	Required	Default	Description
STOP_TIMEOUT			The maximum time (in seconds) in which a stop action script can run. An error message is returned if the script does not complete within the time specified. If you do not specify this attribute or if you specify 0 seconds, then Oracle Clusterware uses the SCRIPT_TIMEOUT value.
TYPE	Yes	None	Must be application.
UPTIME_ THRESHOLD			The value for UPTIME_THRESHOLD represents the length of time that a resource must be up before Oracle Clusterware considers the resource to be stable. By setting a value for the UPTIME_THESHOLD attribute, you can indicate a resource's stability.

Default Profile Locations

Profiles may be located anywhere and need not be on a cluster-visible file system. Oracle RAC provides default locations for profiles as described in the next subsection. The default location for profiles with root privileges on Linux and UNIX systems, or Administrator privileges on Windows systems is the profile subdirectory under the crs directory of the Oracle Clusterware home. The default location for profiles with non-root or non-Administrator privileges is the public subdirectory under the crs directory of the Oracle Clusterware home. The action script must be located in the same directory on all nodes and must be the same file.

Using Entry Points to Manage Resources Using Oracle Clusterware

You can use entry points to specify how to start a resource, stop a resource, and check a resource. You can implement entry points in various ways, such as by using shell or Perl scripts, C++ functions, or Java functions. Oracle Clusterware has the following entry points:

- START—The start (online) entry point brings the resource online.
- STOP—The stop (offline) entry point takes the resource offline.
- CHECK—The check (monitor) entry point monitors the health of a resource.

Example of Using Oracle Clusterware Commands to Create Application Resources

The example in this section creates an application named postman. Oracle Clusterware uses the script /opt/email/bin/crs_postman to start, stop, and monitor whether the application is running (action_script). Oracle Clusterware checks postman every five seconds as specified by the setting for the check_ interval attribute. Oracle Clusterware restarts postman no more than once if it fails. When deciding on which node to place the postman application, Oracle Clusterware considers the value for the optional_resources parameter. If possible, Oracle Clusterware places postman on the same node. Finally, for postman to run, the resource network1 must be running on the same node as specified by the setting for the required_resources parameter. If network1 fails or if it is relocated to another node, then Oracle Clusterware stops or moves the postman application.

Note: Do not use the Oracle Clusterware commands prefixed with `crs_` (except for `crs_stat`) on resources that have names beginning with the prefix `ora` unless either Oracle Support Services ask you to, or unless Oracle has certified you as described in `https://metalink.oracle.com`.

Instead, use the Server Control (SRVCTL) utility on Oracle resources. You can create resources that depend on resources that Oracle has defined. When creating resources, do not use an `ora` prefix in the resource name. This prefix is reserved for Oracle use only.

You can also use the Oracle Clusterware commands to inspect the configuration and status.

Using the crs_profile Command to Create An Application Resource Profile

To create an action profile, you use the `crs_profile` command. Example 5–1 uses the `crs_profile` command to create an application profile for the postman action script that is used to monitor email.

Example 5–1 Using the crs_profile Command to Create an Action Profile

```
$ crs_profile -create postman -t application -B /opt/email/bin/crs_postman \
-d "Email Application" -r network1 -l application2 \
-a postman.scr -o ci=5,ft=2,fi=12,ra=2
```

The contents of the application profile file that the example creates are as follows:

```
NAME=postman
TYPE=application
ACTION_SCRIPT=/oracle/crs/script/postman.scr
ACTIVE_PLACEMENT=0
AUTO_START=always
CHECK_INTERVAL=5
DESCRIPTION=email app
FAILOVER_DELAY=0
FAILURE_INTERVAL=12
FAILURE_THRESHOLD=2
HOSTING_MEMBERS=
OPTIONAL_RESOURCES=application2
PLACEMENT=balanced
REQUIRED_RESOURCES=network1
RESTART_ATTEMPTS=2
SCRIPT_TIMEOUT=60
```

A good example of an action script is the `xclock` script, which is a simple action script that is a default binary on all Linux and UNIX platforms. Example 5–2 shows the contents of the `xclock` action script.

Example 5–2 Action Script Example: xclock

```
#!/bin/bash
# start/stop/check script for xclock example
# To test this change BIN_DIR to the directory where xclock is based
# and set the DISPLAY variable to a server within your network.


BIN_DIR=/usr/X11R6/bin
LOG_DIR=/tmp
```

```
BIN_NAME=xclock
DISPLAY=yourhost.domain.com:0.0
export DISPLAY

if [ ! -d $BIN_DIR ]
then
        echo "start failed"
        exit 2
fi

PID1=`ps -ef | grep $BIN_NAME | grep -v grep | grep -v xclock_app | awk '{ print
$2 }'`

case $1 in
'start')
        if [ "$PID1" != "" ]
        then
           status_p1="running"
        else
           if [ -x $BIN_DIR/$BIN_NAME  ]
           then
             umask 002
             ${BIN_DIR}/${BIN_NAME} & 2>${LOG_DIR}/${BIN_NAME}.log
             status_p1="started"
           else
             echo `basename $0`": $BIN_NAME: Executable not found"
           fi
        fi

         echo "$BIN_NAME: $status_p1"
        ;;

'stop')
        if [ "${PID1}" != "" ]
        then
           kill -9 ${PID1} && echo "$BIN_NAME daemon killed"
        else
           echo "$BIN_NAME: no running Prozess!"
        fi
        ;;
'check')
        if [ "$PID1" != "" ]
        then
           echo "running"
           exit 0
        else
           echo "not running"
           exit 1
        fi
        ;;
*)
        echo "Usage: "`basename $0`" {start|stop|check}"
        ;;
esac
```

See Also: Appendix D, "High Availability Oracle Clusterware Command-Line Reference and C API" for an explanation of the command options

Oracle Clusterware Required Resources List

Oracle Clusterware uses the required resources list, with the placement policy and hosting nodes list, to determine the cluster nodes that are eligible to host an application. Required resources must be ONLINE on the nodes on which the application is running or started.

The failure of a required resource on a hosting node causes Oracle Clusterware to attempt to restart the application on the current node. If RESTART_ATTEMPTS is not set to 0, and if the application cannot start on the current node, then Oracle Clusterware attempts to fail the application over to another node that provides the required resource. Alternatively, Oracle Clusterware stops the application if there is no suitable node. In this case, Oracle Clusterware posts a not restarting event notification.

You can also use required resource lists to start, stop, and relocate groups of interdependent applications when you use the crs_start, crs_stop, or crs_relocate commands with the force (-f) option. In other words, you can configure a set of resources to have other required resources. For example you can configure resources A, B, and C where A and B depend on C. Then if you stop resource C with -force option, this action will stop all three resources. The same is true for the crs_relocate command.

In addition, using the -force option can relocate an online dependency, if necessary, to enable the starting of a resource. For instance, assume that resource VIP_A that has a primary node assignment of node A is running instead on node B. If you perform a crs_start on the instance resource for node A, then this operation will fail. This is because the VIP_A resource is required for the instance on node A and the VIP_A resource is online. However, the resource is on a node where the resource cannot run. Performing a crs_start -f on instance B, however, forces the VIP_A resource to relocate and then start the instance.

Creating VIPs for Applications

If your application is accessed by way of a network, then Oracle recommends that you create a virtual internet protocol address for the application as a dependent resource. In the previous example, network1 is an application VIP address. Create application VIP addresses as follows:

```
crs_profile -create network1 -t application \
-a CRS_home/bin/usrvip \
-o oi=eth0,ov=138.3.83.78,on=255.255.240.0
```

Note: In the case of a user VIP, you must use the usrvip action script that Oracle provides in the *CRS home*/bin directory.

In this example, *CRS_home* is the home directory for the Oracle Clusterware installation. In addition, eth0 is the name of the public network adapter, 138.3.83.78 which resolves by way of DNS to a new hostname that will locate your application regardless of the node on which it is running. Finally, 255.255.240.0 is the netmask for the public IP address. As the oracle user, register the VIP address with Oracle Clusterware as follows:

```
crs_register network1
```

On Linux and UNIX operating systems, the application VIP address script must run as the root user. As the root user, change the owner of the resource as follows:

```
crs_setperm network1 -o root
```

As the `root` user, enable the `oracle` user to run this script:

```
crs_setperm network1 -u user:oracle:r-x
```

As the `oracle` user, start the VIP address as follows:

```
crs_start network1
```

Application Placement Policies

The placement policy specifies how Oracle Clusterware selects a node on which to start an application and where to relocate the application after a node failure. Only cluster nodes on which all of the required resources are available, as listed in an application's profile, are eligible to be considered as hosting nodes for an application. Oracle Clusterware supports the following placement policies:

- `balanced`—Oracle Clusterware favors starting or restarting the application on the node that is currently running the fewest resources. A placement that is based on optional resources is considered first. Next, the host with the fewest resources running is chosen. If no node is favored by these criteria, then any available node is chosen.
- `favored`—Oracle Clusterware refers to the list of nodes in the `HOSTING_MEMBERS` attribute of the application profile. Only cluster nodes that are in this list and that satisfy the resource requirements are eligible for placement consideration. Placement due to optional resources is considered first. If no node is eligible based on optional resources, then the order of the hosting nodes determines which node runs the application. If none of the nodes in the hosting node list are available, then Oracle Clusterware places the application on any available node. This node may or may not be included in the `HOSTING_MEMBERS` list.
- `restricted`—Similar to `favored` except that if none of the nodes on the hosting list are available, then Oracle Clusterware does not start or restart the application. A restricted placement policy ensures that the application never runs on a node that is not on the list, even if you manually relocate it to that node.

You must specify hosting nodes in the `HOSTING_MEMBERS` attribute to use a favored or restricted placement policy. Do not specify hosting nodes in the `HOSTING_MEMBERS` attribute with a balanced placement policy. Otherwise, the application will not validate and you cannot register it. If `ACTIVE_PLACEMENT` is set to `1`, then the placement of the application is reevaluated whenever you add a node to the cluster or if the cluster node restarts. This enables Oracle Clusterware to relocate applications to a preferred node after the node recovers from a failure. To have Oracle Clusterware relocate an application to a preferred node at a time other than when the node rejoins the cluster, use the `REBALANCE` attribute to specify a time at which placement can be reevaluated.

Optional Resources in Placement Decisions

Oracle Clusterware uses optional resources to choose a hosting node based on the number of optional resources that are in an `ONLINE` state on the hosting node. If each node has an equal number of optional resources in an `ONLINE` state, then Oracle Clusterware considers the order of the optional resources as follows:

- Oracle Clusterware compares the state of the optional resources on each node starting at the first resource that you list in the application profile and then proceeds through the list.

- For each consecutive resource in your list, if the resource is ONLINE on one node, then any node that does not have the resource ONLINE is not considered.
- Oracle Clusterware evaluates each resource in the list in this manner until only one node is available to host the resource.
- The maximum number of optional resources is 58.

If this algorithm results in multiple preferred nodes, then the resource is placed on one of these nodes chosen according to its placement policy.

Oracle Clusterware Action Program Guidelines

This section provides the following guidelines for writing Oracle Clusterware action programs that interpret Oracle Clusterware start, stop, and check commands:

- Oracle Clusterware relies on a status code upon exiting from an action program to set the resource state to ONLINE or OFFLINE. On Windows systems, the program should be nonblocking, which may imply a Windows service or a Windows resource that does not block during console interactions.
- Action programs must return a status code to indicate success or failure. For example, on Linux or UNIX systems, if the script is written in Bourne Shell, then the program should issue exit(1) to indicate failure, and exit(0) to indicate success. Similarly, on Windows systems, the action program should return a status of (0) to indicate success and (1) to indicate failure.
- After application failure, Oracle Clusterware calls the action program with a check parameter. The action program replies to Oracle Clusterware with a status of (1) to indicate failure. After receiving this failure status, Oracle Clusterware calls the action program with a stop parameter. It is important that the action program returns a status of (0) to indicate a successful stop of the application, even though the application was not running when the stop request was called.
- Oracle Clusterware sets the resource state to UNKNOWN if an action program's stop entry point fails to exit within the number of seconds in the SCRIPT_TIMEOUT value, or if the action program returns with a code that indicates failure. This may occur during a start, relocation, or stop operation. Ensure that the action program's stop entry point exits with a value that indicates success if the resource is successfully stopped or if the resource is not running.
- When a daemon or service starts, it usually needs to start as a background process, depending on the platform. However, a resource started in this way always returns success during a start attempt. This means that the default scripts cannot detect failures caused by minor errors, such as misspelled command paths.
  - On Linux and UNIX systems, if a resource does not move to the background immediately upon startup, then you can start the application in the background by adding an ampersand (&) to the end of the line that starts the application.
  - On Windows systems, you can use net start to start a service that needs to start in the background.
  - When using commands to start daemons or services in the background, interactively make test runs of the commands used in the script to eliminate errors before using the script with Oracle Clusterware.

How Oracle Clusterware Runs Action Programs

This section describes how Oracle Clusterware runs action programs. The first argument to an action program is the command `start`, `stop`, or `check` depending on which action Oracle Clusterware is running. The second argument is the Oracle Clusterware resource name of the application. This enables a script to determine which instance of the resource that Oracle Clusterware is starting, stopping, or checking.

An action program can retrieve any of its Oracle Clusterware resource attributes from the environment by using `$_CAA_`*`attribute_name`*. For example, `$_CAA_NAME` contains the application name, the second argument to the script, and `$_CAA_HOSTING_MEMBERS` contains its `HOSTING_MEMBERS` attribute.

User Defined Attributes

Oracle Clusterware supports user-defined attributes in Oracle Clusterware applications, which are attributes having names that contain `USR`. User-defined attributes are stored as part of the Oracle Clusterware application profile for the application. You can reference them in an action program using `$_USR_`*`attributename`*.

To add a user-defined attribute, add it to the file *`CRS_home`*`/crs/template/application.tdf` using the following syntax:

```
#
# an example user-defined attribute
#
#!==========================
attribute: USR_EXAMPLE
type: string
switch: -o example
default:
required: no
```

The `attribute` parameter contains the name of the new attribute. The `type` parameter defines the type of the user-defined attribute and can be one of the following:

- `string`
- `boolean`
- `integer`—a numeric attribute
- `positive_integer`—a numeric attribute that must be positive
- `name string`
- `name_list`—a comma-delimited list of names

The `switch` parameter describes how the attribute is specified for the `crs_profile` command. Set the `required` field to `no` for user-defined attributes.

> **Note:** User-defined attribute names that begin with `USR_ORA` are reserved for use by Oracle.

Windows crsuser Program

This section describes the Windows `crsuser` program. The syntax for the `crsuser` command is:

```
crsuser add [domain\]username
```

For example, on a Windows system you could issue the following command as an operating system user that is part of ORA_DBA group and the Local Administrator group:

```
C:\> crsuser add oracledomain\oracluster
```

Provide the user's Windows password. This creates the OracleCRSToken_*user* service that Oracle Clusterware needs to start the Oracle Clusterware resources under the given user ID (when they are not running as the LocalSystem account). You can also use the crsuser command: `remove [domain\]username` to remove a token service and crsuser list to list a registered users.

Using Oracle Clusterware Commands

This section describes how to use the Oracle Clusterware commands under the following topics:

- Registering Application Resources
- Starting Application Resources
- Relocating Applications and Application Resources
- Stopping Applications and Application Resources
- Unregistering Applications and Application Resources
- Displaying Clusterware Application and Application Resource Status Information

Registering Application Resources

Each application that you manage with Oracle Clusterware must have an application profile and the profile must be registered in the OCR. Use the crs_register command to register applications in the OCR. For example, enter the following command to register the mail monitoring application from the previous example:

```
# crs_register postman
```

If you modify a profile, then update the OCR by running the crs_register -u command again.

Starting Application Resources

To start an application resource that is registered with Oracle Clusterware, use the crs_start command. For example:

```
# crs_start postman
```

The following text is an example of the command output:

```
Attempting to start `postman` on node `rac1`
Start of `postman` on node `rac1` succeeded.
```

The application now runs on the node named rac1.

Note: The name of the application resource may or may not be the same as the name of the application.

See Also : Appendix D, "High Availability Oracle Clusterware Command-Line Reference and C API" for examples of Oracle Clusterware command output

The command waits for the amount of time specified by the setting for the SCRIPT_TIMEOUT parameter to receive a notification of success or failure from the action program each time the action program is called. Application resources can be started if they have stopped due to exceeding their failure threshold values. You must register a resource with crs_register before you can start it.

To start and stop the resources, use the crs_start and crs_stop commands. Manual starts or stops outside of Oracle Clusterware can invalidate the resource status. In addition, Oracle Clusterware may attempt to restart a resource on which you perform a manual stop operation.

All required resources must be online on the node where you start the resource. If the resources that the REQUIRED_RESOURCES parameter identifies are offline, then the command crs_start *resource_name* will start the required resources before starting the resource.

Running the crs_start command on a resource sets the resource target value to ONLINE. Oracle Clusterware attempts to change the state to match the target by running the action program with the start parameter. When a resource is running, both the target state and current state are ONLINE.

Starting an Application on an Unavailable Node

When starting an application on a cluster node that is unavailable, crs_start can give indeterminate results. In this scenario, the start section of the action program is run, but the cluster node fails before notification of the start is displayed on the command line. The crs_start command returns a failure with the error Remote start for *resource_name* failed on node *node_name*. The application is actually ONLINE but fails over to another node making the application appear as though it were started on the incorrect node.

If a cluster node fails while you are starting a resource on that node, then check the state of the resource on the cluster by using the crs_stat command to determine the state of that resource.

Relocating Applications and Application Resources

Use the crs_relocate command to relocate applications and application resources. For example, to relocate the mail monitoring application to the node known as rac2, enter the following command:

```
# crs_relocate postman -c rac2
```

Each time that the action program is called, the crs_relocate command waits for the duration identified by the value for the SCRIPT_TIMEOUT parameter to receive notification of success or failure from the action program. A relocation attempt fails if:

- The application has required resources that are ONLINE on the initial node
- Applications that require the specified resource are ONLINE on the initial node

To relocate an application and its required resources, use the -f option with the crs_relocate command. Oracle Clusterware relocates or starts all resources that are required by the application regardless of their state.

Stopping Applications and Application Resources

To stop applications and application resources, use the `crs_stop` command. Immediately after the `crs_stop` command completes, the application status converts to `OFFLINE`. Because Oracle Clusterware always attempts to match a resource's state to its target, the Oracle Clusterware subsystem stops the application. The following example stops the mail application from the example:

```
# crs_stop postman
```

The following text is an example of the command output:

```
Attempting to stop `postman` on node `rac1`
Stop of `postman` on node `rac1` succeeded.
```

You cannot stop an application if the application is a required resource for another online application unless you use the force (`-f`) option. If you use the `crs_stop -f` *`resource_name`* command on an application that is required by other resources and if those resources are online, then Oracle Clusterware stops the application. In addition, all of the resources that require that application that are online are also stopped.

Note: Oracle Clusterware can only stop applications and application resources. Oracle Clusterware cannot stop network, tape, or media changer resources.

Managing Automatic Oracle Clusterware Resource Operations for Action Scripts

The following section explains additional information for controlling how Oracle Clusterware manages restarts. You can prevent Oracle Clusterware from automatically restarting a resource by setting several action program attributes. You can also control how Oracle Clusterware manages the restart counters for your action programs. In addition, you can customize the timeout values for the `start`, `stop`, and `check` actions that Oracle Clusterware performs on action scripts. These topics are described under the following headings:

- Preventing Automatic Restarts
- Automatically Manage Restart Attempts Counter for Resources
- Implications of Restart and Timeout Features for Previous Releases

Preventing Automatic Restarts

When a node stops and restarts, Oracle Clusterware starts the resources as soon as the node starts. This may not be desirable because resource startup might fail if system components on which the resource depends, such as a volume manager or a file system, are not running. This is especially true if Oracle Clusterware does not manage the system components on which the resource depends. To manage automatic restarts, you can use the `AUTO_START` attribute to specify whether Oracle Clusterware should automatically start a resource when a node restarts.

Valid `AUTO_START` values are:

- `always`—Causes the resource to restart when the node restarts regardless of the resource's state when the node stopped.
- `restore`—Does not start the resource at restart time if it was in an offline state, such as `STATE=OFFLINE`, `TARGET=OFFLINE`, when the node stopped. The

resource is restored to its state when the node went down. The resource is started only if it was online before and not otherwise.

- `never`—Oracle Clusterware never restarts the resource regardless of the resource's state when the node stopped.

> **Note:** Oracle only supports lower-case values for `always`, `restore`, and `never`.

Automatically Manage Restart Attempts Counter for Resources

When a resource fails, Oracle Clusterware restarts the resource for only the number of times specified in the profile attribute `RESTART_ATTEMPTS` regardless of how often the resource fails. The `CRSD` process maintains an internal counter to track how often a resource has been restarted. There is a mechanism by which Oracle Clusterware can automatically manage the restart attempts counter based on the stability of a resource. Use the `UPTIME_THESHOLD` attribute to indicate resource stability.

You can specify the time for the `UPTIME_THRESHOLD` attribute in different units of measure, such as seconds (`s`), minutes (`m`), hours (`h`), days (`d`) or weeks (`w`). Examples of valid values for this attribute are: `7d` for seven days, `5h` for five hours, `180m` for 180 minutes, and so on. Specify the time period as a numeric value and the unit of measure as the last character, `s`, `m`, `h`, `d`, or `w`.

After the time period that you have indicated by the setting for `UPTIME_THRESHOLD` has elapsed, Oracle Clusterware resets the value for `RESTART_COUNTS` to 0. Oracle Clusterware can alert you when the value for `RESTART_COUNT` reaches the value that you have set for `RESTART_ATTEMPTS`.

> **Note:** Oracle Clusterware writes an alert to the CRSD log file when the value for `RESTART_COUNT` reaches the value that you have set for `RESTART_ATTEMPTS`. Oracle Clusterware does not write this message to the alert file.

RESTART_ATTEMPTS and RESTART_COUNT Failure Scenarios Some of the failure scenarios for the `RESTART_ATTEMPTS` and `RESTART_COUNT` attributes are:

- When a resource keeps restarting—The resource does not meet the uptime threshold criteria and it will be stopped after restarting for the number of attempts set by the value for `RESTART_ATTEMPTS`.
- When a node fails or restarts—Oracle Clusterware resets the value for `RESTART_COUNTER` to `0` either when the resource relocates or when it restarts on the same node.
- If the `crsd` process fails—Because both `RESTART_COUNT` and `RESTART_ATTEMPTS` are stored in OCR, the behavior is not affected.

Implications of Restart and Timeout Features for Previous Releases

If you have an installation that is prior to Oracle Database 11*g*, add the release 1 (11.1) attributes to your profiles by doing one of the following:

- Modify your resources with the Oracle Database 11*g* attributes and reregister the resources. This causes the separate timeouts to be effective.

- Do not modify your resources. This retains the pre-release behavior of Oracle Database 11*g* using the value in `SCRIPT_TIMEOUT` as the timeout for all start, stop, and check actions.

Unregistering Applications and Application Resources

To unregister an application or an application resource, use the `crs_unregister` command. You cannot unregister an application or resource that is `ONLINE` or required by another resource. The following example unregisters the mail application:

```
# crs_unregister postman
```

The unregistration process frees space on the OCR. Additionally, run the `crs_unregister` command as a clean-up step when a resource is no longer managed by Oracle Clusterware. Generally, you should unregister all permanently stopped applications.

Displaying Clusterware Application and Application Resource Status Information

To display status information about applications and resources that are on cluster nodes, use the `crs_stat` command. The following example displays the status information for the `postman` application:

```
# crs_stat postman
NAME=postman
TYPE=application
TARGET=ONLINE
STATE=ONLINE on rac2
```

Enter the following command to view information about all applications and resources in tabular format:

```
# crs_stat -t
```

The following text is an example of the command output:

```
Name         Type        Target      State     Host
------------------------------------------------------------
cluster_lockd application ONLINE    ONLINE     rac2
dhcp          application OFFLINE   OFFLINE
```

Enter the following command to determine:

- How many times an application resource has been restarted
- How many times an application resource has failed within the failure interval
- The maximum number of times that an application or resource can be restarted or fail
- The target state of the application or resource and the normal status information

```
# crs_stat -v
```

To view verbose content in tabular format, enter the following command:

```
# crs_stat -v -t
```

The following text is an example of the command output:

```
Name          Type       R/RA  F/FT   Target    State    Host
----------------------------------------------------------------------
cluster_lockd application 0/30  0/0   ONLINE    ONLINE   rac2
```

```
dhcp          application 0/1   0/0   OFFLINE   OFFLINE
named         application 0/1   0/0   OFFLINE   OFFLINE
network1      application 0/1   0/0   ONLINE    ONLINE   rac1
```

Enter the following command to view the application profile information that is stored in the OCR:

```
# crs_stat -p
```

The following text is an example of the command output:

```
NAME=cluster_lockd
TYPE=application
ACTION_SCRIPT=cluster_lockd.scr
ACTIVE_PLACEMENT=0
AUTO_START=restore
CHECK_INTERVAL=5
DESCRIPTION=Cluster lockd/statd
FAILOVER_DELAY=30
FAILURE_INTERVAL=60
FAILURE_THRESHOLD=1
REBALANCE=
HOSTING_MEMBERS=
OPTIONAL_RESOURCES=
PLACEMENT=balanced
REQUIRED_RESOURCES=
RESTART_ATTEMPTS=2
SCRIPT_TIMEOUT=60 ...
```

See the `crs_stat` command for more information.

> **See Also:** Appendix D, "High Availability Oracle Clusterware Command-Line Reference and C API" for detailed information about Oracle Clusterware commands

A

Cluster Verification Utility Reference

Oracle provides Cluster Verification Utility (CVU) to perform system checks in preparation for installation, patch updates, or other system changes. Learning how to use CVU can ensure that you have completed the required system configuration and preinstallation steps so that your installation, update, or patch operation completes successfully.

This appendix describes the CVU under the following topics:

- Using the Cluster Verification Utility
- Cluster Verification Utility Requirements
- Understanding CVU Commands, Help, Output, and Nodelist Shortcuts
- Performing Various CVU Tests
- Known Issues for the Cluster Verification Utility

> **See Also:** Your platform-specific Oracle Clusterware and Oracle RAC installation guide for information about how to manually install CVU

Using the Cluster Verification Utility

The CVU can verify the primary cluster components during an operational phase or stage. A component can be basic, such as free disk space, or it can be complex, such as checking Oracle Clusterware integrity. For example, CVU can verify multiple Oracle Clusterware subcomponents across Oracle Clusterware layers. Additionally, CVU can check disk space, memory, processes, and other important cluster components. A stage could be, for example, database installation, for which CVU can verify whether your system meets the criteria for an Oracle RAC installation. Other stages include the initial hardware setup and the establishing of system requirements through the fully operational cluster setup.

When verifying stages, CVU uses entry and exit criteria. In other words, each stage has entry criteria that define a specific set of verification tasks to be performed before initiating that stage. This check prevents you from beginning a stage, such as installing Oracle Clusterware, unless you meet the Oracle Clusterware stage's prerequisites.

The exit criteria for a stage define another set of verification tasks that you need to perform after the completion of the stage. Post-checks ensure that the activities for that stage have been completed. Post-checks identify stage-specific problems before they propagate to subsequent stages.

The node list that you use with CVU commands should be a comma-delimited list of host names without a domain. The CVU ignores domains while processing node lists.

If a CVU command entry has duplicate node entries after removing domain information, then CVU eliminates the duplicate node entries. Wherever supported, you can use the `-n all` option to verify all of your cluster nodes that are part of a specific Oracle RAC installation. You do not have to be the `root` user to use the CVU and the CVU assumes that the current user is the `oracle` user.

Note: The CVU only supports an English-based syntax and English online help.

For network connectivity verification, the CVU discovers all of the available network interfaces if you do not specify an interface on the CVU command line. For storage accessibility verification, the CVU discovers shared storage for all of the supported storage types if you do not specify a particular storage identification on the command line. The CVU also discovers the Oracle Clusterware home if one is available.

Run the CVU command-line tool using the `cluvfy` command. Using `cluvfy` does not adversely affect your cluster environment or your installed software. You can run `cluvfy` commands at any time, even before the Oracle Clusterware installation. In fact, the CVU is designed to assist you as soon as your hardware and operating system are operational. If you run a command that requires Oracle Clusterware on a node, then the CVU reports an error if Oracle Clusterware is not yet installed on that node.

You can enable tracing by setting the environment variable `SRVM_TRACE` to true. For example, in `tcsh` an entry such as `setenv SRVM_TRACE true` enables tracing. The CVU trace files are created in the `CV_HOME/cv/log` directory. Oracle automatically rotates the log files and the most recently created log file has the name `cvutrace.log.0`. You should remove unwanted log files or archive them to reclaim disk place if needed. The CVU does not generate trace files unless you enable tracing.

Cluster Verification Utility Requirements

The CVU requirements are:

- At least 30MB free space for the CVU software on the node from which you run the CVU
- A location for the current JDK, Java 1.4.1 or later
- A work directory with at least 25MB free space on each node

Note: When using the CVU, the CVU attempts to copy any needed information to the CVU work directory. Make sure that the CVU work directory exists on all of the nodes in your cluster database and that the directory on each node has write permissions established for the CVU user. Set this directory using the `CV_DESTLOC` environment variable. If you do not set this variable, then the CVU uses `/tmp` as the work directory on Linux and UNIX systems, and `C:\temp` on Windows systems.

Understanding CVU Commands, Help, Output, and Nodelist Shortcuts

This section describes the following Cluster Verification Utility topics:

- Using CVU Help
- Verbose Mode and UNKNOWN Output

Using CVU Help

The `cluvfy` commands have context sensitive help that shows their usage based on the command-line arguments that you enter. For example, if you enter `cluvfy`, then the CVU displays high-level generic usage text describing the stage and component syntax. If you enter `cluvfy comp -list`, then the CVU shows the valid components with brief descriptions about each of them. If you enter `cluvfy comp -help`, then the CVU shows detailed syntax for each of the valid component checks. Similarly, `cluvfy stage -list` and `cluvfy stage -help` display valid stages and their syntax for their checks respectively.

If you enter an invalid CVU command, then the CVU shows the correct usage for that command. For example, if you type `cluvfy stage -pre dbinst`, then CVU shows the correct syntax for the precheck commands for the `dbinst` stage. Enter the `cluvfy -help` command to see detailed CVU command information.

Verbose Mode and UNKNOWN Output

Although by default the CVU reports in nonverbose mode by only reporting the summary of a test, you can obtain detailed output by using the `-verbose` argument. The `-verbose` argument produces detailed output of individual checks and where applicable shows results for each node in a tabular layout.

If a `cluvfy` command responds with `UNKNOWN` for a particular node, then this is because the CVU cannot determine whether a check passed or failed. The cause of this could be a loss of reachability or the failure of user equivalence to that node. The cause could also be any system problem that was occurring on that node at the time that CVU was performing a check.

If you run the CVU using the `-verbose` argument and the CVU responds with `UNKNOWN` for a particular node, then this is because the CVU cannot determine whether a check passed or failed. The following is a list of possible causes for an `UNKNOWN` response:

- The node is down
- Executables that the CVU requires are missing in *`CRS_home`*`/bin` or the *`Oracle home`* directory
- The user account that ran the CVU does not have privileges to run common operating system executables on the node
- The node is missing an operating system patch or a required package
- The node has exceeded the maximum number of processes or maximum number of open files, or there is a problem with IPC segments, such as shared memory or semaphores

Cluster Verification Utility Nodelist Shortcuts

You can use the following nodelist shortcuts:

To provide the CVU a list of all of the nodes of a cluster, enter `-n all`. CVU attempts to obtain the node list in the following order:

1. If vendor clusterware is available, then the CVU selects all of the configured nodes from the vendor clusterware using the `lsnodes` utility.

2. If Oracle Clusterware is installed, then the CVU selects all of the configured nodes from Oracle Clusterware using the `olsnodes` utility.
3. If neither the vendor nor Oracle Clusterware is installed, then the CVU searches for a value for the `CV_NODE_ALL` key in the configuration file.
4. If vendor and Oracle Clusterware are not installed and no key named `CV_NODE_ALL` exists in the configuration file, then the CVU searches for a value for the `CV_NODE_ALL` environmental variable.

If you have not set this variable, then the CVU reports an error.

To provide a partial node list, you can set an environmental variable and use it in the CVU command. For example, on Linux or UNIX systems you can enter:

```
setenv MYNODES node1,node3,node5
cluvfy comp nodecon -n $MYNODES [-verbose]
```

Cluster Verification Utility Configuration File

You can use the CVU configuration file to define specific inputs for the execution of the CVU. The path for the configuration file is `CV_HOME/cv/admin/cvu_config`. You can modify this using a text editor. The inputs to the tool are defined in the form of key entries. You must follow these rules when modifying the CVU configuration file:

- Key entries have the syntax `name=value`
- Each key entry and the value assigned to the key only defines one property
- Lines beginning with the number sign (`#`) are comment lines and are ignored
- Lines that do not follow the syntax `name=value` are ignored

The following is the list of keys supported by CVU:

- `CV_NODE_ALL`—If set, it specifies the list of nodes that should be picked up when Oracle Clusterware is not installed and a `-n` all option has been used in the command line. By default, this entry is commented out.
- `CV_RAW_CHECK_ENABLED`—If set to `TRUE`, it enables the check for accessibility of shared disks on RedHat release 3.0. This shared disk accessibility check requires that you install a `cvuqdisk` rpm on all of the nodes. By default, this key is set to `TRUE` and shared disk check is enabled.
- `CV_XCHK_FOR_SSH_ENABLED`—If set to `TRUE`, it enables the X-Windows check for verifying user equivalence with ssh. By default, this entry is commented out and X-Windows check is disabled.
- `ORACLE_SRVM_REMOTESHELL`—If set, it specifies the location for `ssh/rsh` command to override the CVU default value. By default, this entry is commented out and the tool uses `/usr/sbin/ssh` and `/usr/sbin/rsh`.
- `ORACLE_SRVM_REMOTECOPY`—If set, it specifies the location for the `scp` or `rcp` command to override the CVU default value. By default, this entry is commented out and CVU uses `/usr/bin/scp` and `/usr/sbin/rcp`.

If CVU does not find a key entry defined in the configuration file, then the CVU searches for the environment variable that matches the name of the key. If the environment variable is set, then the CVU uses its value, otherwise the CVU uses a default value for that entity.

Performing Various CVU Tests

You can perform the following tests using CVU as described under the following topics:

- Cluster Verification Utility System Requirements Verifications
- Cluster Verification Utility Storage Verifications
- Cluster Verification Utility Connectivity Verifications
- Cluster Verification Utility User and Permissions Verifications
- Cluster Verification Utility Node Comparisons and Verifications
- Cluster Verification Utility Installation Verifications
- Cluster Verification Utility Oracle Clusterware Component Verifications
- Cluster Verification Utility Cluster Integrity Verifications
- Cluster Verification Utility Argument and Option Definitions

> **See Also:** Table A–1 for details about the arguments and options used in the following CVU examples

Cluster Verification Utility System Requirements Verifications

To verify the minimal system requirements on the nodes prior to installing Oracle Clusterware or Oracle RAC, use the `sys` component verification command as follows:

```
cluvfy comp sys [ -n node_list ] -p { crs | database } }  [-r { 10gR1 | 10gR2 |
 11gR1} ] [ -osdba osdba_group ] [ -orainv orainventory_group ] [-verbose]
```

To check the system requirements for installing Oracle RAC, use the `-p database` argument, and to check the system requirements for installing Oracle Clusterware, use the `-p crs` argument. To check the system requirements for installing Oracle Clusterware or Oracle RAC from Oracle Database 11*g* release 1 (11.1), use the `-r 11gR1` argument. For example, verify the system requirements for installing Oracle Clusterware on the cluster nodes known as `node1`,`node2` and `node3` by running the following command:

```
cluvfy comp sys -n node1,node2,node3 -p crs -verbose
```

Cluster Verification Utility Storage Verifications

To verify whether storage is shared among the nodes in your cluster database or to identify all of the storage that is available on the system and can be shared across the cluster nodes, use the component verification command `ssa` as follows:

```
cluvfy comp ssa [ -n node_list ] [ -s storageID_list ] [-verbose]
```

> **See Also:** "Known Issues for the Cluster Verification Utility" on page A-9 for the types of storage that CVU supports

For example, discover all of the shared storage systems available on your system by running the following command:

```
cluvfy comp ssa  -n all -verbose
```

You can verify the accessibility of a specific storage location, such as `/dev/sda`, across the cluster nodes by running the following command:

```
cluvfy comp ssa  -n all -s /dev/sda
```

To verify whether a certain amount of free space is available on a specific location in the nodes of your cluster database, use the component verification command `space`.

```
cluvfy comp space  [ -n node_list ] -l storage_location -z disk_space {B|K|M|G}
 [-verbose]
```

For example, you can verify the availability of at least 2 GB of free space at the location `/home/dbadmin/products` on all of the cluster nodes by running the following command:

```
cluvfy comp space  -n all -l  / home/dbadmin/products  -z 2G -verbose
```

To verify the integrity of your Oracle Cluster File System (OCFS) on platforms on which OCFS is available, use the component verification command `cfs` as follows:

```
cluvfy comp cfs  [ -n node_list ] -f file_system [-verbose]
```

For example, you can verify the integrity of the cluster file system `/oradbshare` on all of the nodes by running the following command:

```
cluvfy comp cfs -f /oradbshare -n all -verbose
```

Note: The sharedness check for the file system is supported for Oracle Cluster File System version 1.0.14 or higher.

Cluster Verification Utility Connectivity Verifications

To verify the cluster nodes can be reached from the local node or from any other cluster node, use the component verification command `nodereach` as follows:

```
cluvfy comp nodereach -n node_list [ -srcnode node ]  [-verbose]
```

To verify the connectivity between the cluster nodes through all of the available network interfaces or through specific network interfaces, use the component verification command `nodecon` as follows:

```
cluvfy comp nodecon -n node_list [ -i interface_list ]  [-verbose]
```

Use the `nodecon` command without the `-i` option as follows to use CVU to:

- Discover all of the network interfaces that are available on the cluster nodes
- Review the interfaces' corresponding IP addresses and subnets
- Obtain the list of interfaces that are suitable for use as VIPs and the list of interfaces to private interconnects
- Verify the connectivity between all of the nodes through those interfaces

```
cluvfy comp nodecon -n all [-verbose]
```

You can run this command in verbose mode to identify the mappings between the interfaces, IP addresses, and subnets. To verify the connectivity between all of the nodes through specific network interfaces, use the `comp nodecon` command with the `-i` option. For example, you can verify the connectivity between the nodes `node1`,`node2`, and `node3`, through interface `eth0` by running the following command:

```
cluvfy comp nodecon -n node1,node2,node3 -i eth0 -verbose
```

Cluster Verification Utility User and Permissions Verifications

To verify user accounts and administrative permissions-related issues, use the component verification command `admprv` as follows:

```
cluvfy comp admprv  [ -n node_list ]  [-verbose]
                | -o user_equiv [-sshonly]
                | -o crs_inst  [-orainv orainventory_group ]
                | -o db_inst  [-orainv orainventory_group ] [-osdba osdba_group ]
                | -o db_config  -d oracle_home
```

To verify whether user equivalence exists on specific nodes, use the `-o user_equiv` argument. On Linux and UNIX platforms, this command verifies user equivalence first using `ssh` and then using `rsh`, if the `ssh` check fails. To verify the equivalence only through `ssh`, use the `-sshonly` option. By default, the equivalence check does not verify X-Windows configurations, such as whether you have disabled X-forwarding, whether you have the proper setting for the `DISPLAY` environment variable, and so on.

To verify X-Windows aspects during user equivalence checks, set the `CV_XCHK_FOR_SSH_ENABLED` key to `TRUE` in the configuration file that resides in the path `CV_HOME/cv/admin/cvu_config` before you run the `admprv -o user_equiv` command. Use the `-o crs_inst` argument to verify whether you have permissions to install Oracle Clusterware.

You can use the `-o db_inst` argument to verify the permissions that are required for installing Oracle RAC and the `-o db_config` argument to verify the permissions that are required for creating an Oracle RAC database or for modifying an Oracle RAC database's configuration. For example, you can verify user equivalence for all of the nodes by running the following command:

```
cluvfy comp admprv  -n all -o user_equiv -verbose
```

On Linux and UNIX platforms, this command verifies user equivalence by first using ssh and then using rsh if the ssh check fails. To verify the equivalence only through ssh, use the `-sshonly` option. By default, the equivalence check does not verify X-Windows configurations, such as when you have disabled X-forwarding with the setting of the `DISPLAY` environment variable. To verify X-Windows aspects during user equivalence checks, set the `CV_XCHK_FOR_SSH_ENABLED` key to `TRUE` in the configuration file `CV_HOME/cv/admin/cvu_config` before you run the `admprv -o user_equiv` command.

To verify the existence of node applications, namely VIP, ONS and GSD, on all of the nodes, use the component `nodeapp` command:

```
cluvfy comp nodeapp  [ -n node_list ]  [-verbose]
```

Cluster Verification Utility Node Comparisons and Verifications

Use the component verification `peer` command to compare the nodes as follows:

```
cluvfy comp peer [ -refnode node ] -n node_list [-r { 10gR1 | 10gR2 | 11gR1} ]
 [ -orainv orainventory_group ] [ -osdba osdba_group ]  [-verbose]
```

The following command lists the values of several preselected properties on different nodes from Oracle Database 11*g* release 1 (11.1):

```
cluvfy comp peer -n node_list [-r 11gR1] [-verbose]
```

You can also use the `comp peer` command with the `-refnode` argument to compare the properties of other nodes against the reference node.

Cluster Verification Utility Installation Verifications

To verify whether your system meets all of the criteria for an Oracle Clusterware installation, use the `-pre crsinst` command for the Oracle Clusterware installation stage as follows:

```
cluvfy stage -pre crsinst -n node_list
 [ -c ocr_location ] [-r { 10gR1 | 10gR2 | 11gR1} ][ -q voting_disk ]
 [ -osdba osdba_group ]
 [ -orainv orainventory_group ]  [-verbose]
```

After you have completed phase one, verify that Oracle Clusterware is functioning properly before proceeding with phase two of your Oracle RAC installation by running the `-post` crsinst command for the Oracle Clusterware installation stage:

```
cluvfy stage -post crsinst -n node_list [-verbose]
```

To verify whether your system meets all of the criteria for an Oracle RAC installation, use the `pre dbinst` command for the Database Installation stage:

```
cluvfy stage -pre dbinst -n node_list [-r { 10gR1 | 10gR2 | 11gR1} ]
 [ -osdba osdba_group ]
 [ -orainv orainventory_group ]  [-verbose]
```

To verify whether your system meets all of the criteria for creating a database or for making a database configuration change, use the `pre dbcfg` command for the Database Configuration stage:

```
cluvfy stage -pre dbcfg -n node_list -d oracle_home [-verbose]
```

Cluster Verification Utility Cluster Integrity Verifications

To check the integrity of your entire cluster, which means to verify that all of the nodes in the cluster have the same view of the cluster configuration, use the component verification command `comp clu`, as follows:

```
cluvfy comp clu
```

Cluster Verification Utility Oracle Clusterware Component Verifications

To verify the integrity of all of the Oracle Clusterware components, use the component verification `comp crs` command:

```
cluvfy comp crs  [ -n node_list ]  [-verbose]
```

To verify the integrity of each individual Cluster Manager subcomponent, use the component verification command `comp clumgr`:

```
cluvfy comp clumgr  [ -n node_list ]  [-verbose]
```

To verify the integrity of the Oracle Cluster Registry, use the component verification command `comp ocr`:

```
cluvfy comp ocr  [ -n node_list ] [-verbose]
```

Cluster Verification Utility Argument and Option Definitions

Table A–1 describes the CVU arguments and options used in the previous examples:

Table A–1 Cluster Verification Utility Arguments and Options

Argument or Option	Definition
`-n` *`node_list`*	The comma-delimited list of nondomain qualified node names on which the test should be conducted. If `all` is specified, then all of the nodes in the cluster will be used for verification.
`-i` *`interface_list`*	The comma-delimited list of interface names.
`-f` *`file_system`*	The name of the file system.
`-s storageID_list`	The comma-delimited list of storage identifiers.
`-l storage_location`	The storage path.
`-z disk_space`	The required disk space, in units of bytes (B), kilobytes (K), megabytes (M), or gigabytes (G).
`-osdba osdba_group`	The name of the OSDBA group. The default is `dba`.
`-orainv orainventory_group`	The name of the Oracle inventory group. The default is `oinstall`.
`-verbose`	Makes CVU print detailed output.
`-o user_equiv`	Checks user equivalence between the nodes.
`-sshonly`	Check user equivalence for ssh setup only.
`-o crs_inst`	Checks administrative privileges for installing Oracle Clusterware.
`-o db_inst`	Checks administrative privileges for installing Oracle RAC.
`-o db_config`	Checks administrative privileges for creating or configuring a database.
`-refnode`	The node that will be used as a reference for checking compatibility with other nodes.
`-srcnode`	The node from which the reachability to other nodes should be checked.
`-r { 10gR1 | 10gR2 | 11gR1}`	The release of the Oracle Database for which the requirements for installation of Oracle Clusterware or Oracle RAC are to be verified. If this option is not specified, then Oracle Database 11*g* release 1 (11.1) is assumed.

Known Issues for the Cluster Verification Utility

This section describes the following known limitations for CVU:

- Database Versions Supported by Cluster Verification Utility
- Linux Shared Storage Accessibility (ssa) Check Reports Limitations
- Shared Disk Discovery on Red Hat Linux

Database Versions Supported by Cluster Verification Utility

The current CVU release supports only Oracle Database 10*g* or higher, Oracle RAC, and Oracle Clusterware and CVU is not backward compatible. In other words, CVU cannot check or verify Oracle Database products prior to Oracle Database 10*g*.

Linux Shared Storage Accessibility (ssa) Check Reports Limitations

The current release of `cluvfy` has the following limitations on Linux regarding shared storage accessibility check.

- Currently NAS storage (r/w, no attribute caching) and OCFS (version 1.0.14 or higher) are supported.

- For sharedness checks on NAS, `cluvfy` commands require you to have write permission on the specified path. If the `cluvfy` user does not have write permission, `cluvfy` reports the path as `not shared`.

Shared Disk Discovery on Red Hat Linux

To perform discovery and shared storage accessibility checks for SCSI disks on Red Hat Linux 3.0 (or higher) and SUSE Linux Enterprise Server, CVU requires the CVUQDISK package. If you attempt to use CVU and the CVUQDISK package is not installed on all of the nodes in your Oracle RAC environment, then CVU responds with an error.

Perform the following procedure to install the CVUQDISK package:

1. Login as the `root` user.
2. Copy the rpm, `cvuqdisk-1.0.1-1.rpm`, to a local directory. You can find this rpm in the `rpm` subdirectory of the top-most directory in the Oracle Clusterware installation media. For example, you can find `cvuqdisk-1.0.1-1.rpm` in the directory `/`*`mountpoint`*`/clusterware/rpm/` where *`mountpoint`* is the mounting point for the disk on which the directory is located.
3. Set the environment variable to a group that should own the CVUQDISK package binaries. If `CVUQDISK_GRP` is not set, then by default the `oinstall` group is the owner's group.
4. Determine whether previous versions of the CVUQDISK package are installed by running the command `rpm -q cvuqdisk`. If you find previous versions of the CVUQDISK package, then remove them by running the command `rpm -e cvuqdisk` *`previous_version`* where *`previous_version`* is the identifier of the previous CVUQDISK version.
5. Install the latest CVUQDISK package by running the command `rpm -iv cvuqdisk-1.0.1-1.rpm`.

B

OLSNODES Command Reference

This appendix describes the syntax and command options for the OLSNODES command. The OLSNODES command provides the list of nodes and other information for all nodes participating in the cluster. The syntax for the OLSNODES command is:

```
olsnodes [-n] [-i] [-l] [-v] [-g] [-p]
```

If you issue the OLSNODES command without any command parameters, the command returns a listing of the nodes in the cluster:

```
[user1@node1 myoracle]# olsnodes
node1
node2
node3
node4
```

Table B–1 describes the options you can include on the OLSNODES command to obtain additional cluster-related information.

Table B–1 OLSNODES Command Options

Option	Description
-g	Logs cluster verification information with more details.
-i	Lists all nodes participating in the cluster and includes the Virtual Internet Protocol (VIP) address assigned to each node.
-l	Displays the local node name.
-n	Lists al nodes participating in the cluster and includes the assigned node numbers.
-p	Lists all nodes participating in the cluster and includes the private interconnect assigned to each node.
-v	Logs cluster verification information in verbose mode.

C

Oracle Interface Configuration (OIFCFG) Command Reference

The Oracle Interface Configuration (OIFCFG) command-line interface helps you to define and administer network interfaces. You can issue OIFCFG commands in single-instance and Oracle Clusterware environments to:

- Allocate and deallocate network interfaces to components
- Direct components to use specific network interfaces
- Retrieve component configuration information

This appendix contains the following main topics:

Starting the OIFCFG Command-Line Interface

Before you invoke OIFCFG, ensure that you have started Oracle Clusterware on at least the local node and preferably on all nodes if you intend to include the `-global` option on the command.

You invoke OIFCFG from the *`CRS_home`*`/bin/` directory as the CRS user who installed the Oracle Clusterware software.

For example:

```
% ./oifcfg
```

Issue the `oifcfg -help` command to display online help for OIFCFG.

```
oifcfg iflist
  oifcfg setif {-node <nodename> | -global} {<if_name>/<subnet>:<if_type>}...
  oifcfg getif [-node <nodename> | -global] [ -if <if_name>[/<subnet>] [-type <if_
type>]]
  oifcfg delif [-node <nodename> | -global] [<if_name>[/<subnet>]]
  oifcfg [-help]

 <nodename> - name of the host, as known to a communications network
 <if_name>  - name by which the interface is configured in the system
 <subnet>   - subnet address of the interface
 <if_type>  - type of the interface {cluster_interconnect|public|storage}
```

Summary of the OIFCFG Usage

This section contains the following topics:

OIFCFG Command Format

```
oifcfg iflist
oifcfg setif {-node nodename | -global} {if_name/subnet:if_type} ...
oifcfg getif [-node nodename | -global] [-if if_name [/subnet] [-type if_type]]
oifcfg delif [-node nodename | -global] [if_name [/subnet]]
oifcfg [-help]
```

OIFCFG Commands

You can enter any of the OIFCFG commands listed in Table C–1.

Table C–1 OIFCFG Commands

Command	Description
`oifcfg iflist`	Shows the available interfaces that you can configure with `setif`. The `iflist` command queries the operating system to find which network interfaces are present on this node.
`oifcfg setif`	Sets an interface type (public or cluster interconnect) for an interface.
`oifcfg getif`	Displays the interfaces for which an interface type has been defined with the `setif` command, along with the type for that interface.
`oifcfg delif`	Deletes the stored network configuration for global or node-specific interfaces.

OIFCFG Command Parameters

This section lists the parameters for the OIFCFG commands. Note that some of the parameters are optional, depending on which command you issue.

-node <*nodename*>
The name of the Oracle Clusterware node as listed in the output from the `olsnodes` command. The `OLSNODES` command is described in Appendix B, "OLSNODES Command Reference".

-global
A network interface can be stored as a *global interface* (as reported by the `iflist` command) or as a *node-specific interface*:

- An interface is stored as a global interface when all of the nodes of an Oracle RAC cluster have the same interface connected to the same subnet. The global interface (and configuring all nodes with the same network interface for each public subnet and the same network interface for each private subnet) is not only the recommended configuration, but it is also the default installation configuration.
- An interface can be stored as a node-specific (local) interface.

Note: Oracle currently does not support having different network interfaces for each node in the cluster. The best practice is to configure all nodes with the same network interface for each public subnet and the same network interface for each private subnet. See "Changing Network Addresses" on page 2-11 for information about changing interface names.

-if <*if_name*>
The name by which the interface is configured in the system.

subnet
The subnet number of the interface.

-type <*if_type*>
The type of interface: public or cluster interconnect.

-help
Display online help for OIFCFG commands.

OIFCFG Usage Notes

- A network interface specification takes the following form:

  interface_name/*subnet*:*interface_type*

  The specification uniquely identifies the network interface using the:

  – Interface name

  – Associated subnet

  – Interface type

    The interface type indicates the purpose for which the network is configured. The supported interface types are:

    * Public—An interface that can be used for communication with components external to Oracle RAC instances, such as Oracle Net and Virtual Internet Protocol (VIP) addresses.

    * Cluster_interconnect—A private interface used for the cluster interconnect to provide interinstance or Cache Fusion[1] communication.

    If you set the interface type to cluster_interconnect, it affects instances as they start up and changes do not take effect until you restart the instances.

  For example, the following specification identifies `qfe0` as a cluster interconnect located at the address 204.152.65.0:

  ```
  qfe0/204.152.65.0:cluster_interconnect
  ```

- The Oracle Universal Installer (OUI) uses OIFCFG to identify and display available interfaces.
- The effect of changing the interface names depends on which name you are changing, and whether or not you are also changing the IP address. In cases where you change only the interface names, the ramifications are minor. If you change

[1] Cache Fusion is a diskless cache coherency mechanism that provides copies of blocks directly from a holding instance's memory cache to a requesting instance's memory cache.

the name for the public interface that is stored in the OCR, you must modify the nodeapps for each node. Therefore, you must stop the nodeapps for this change to take effect.

- You must restart Oracle Clusterware on all members of the cluster when you make global changes. For local changes, you only need to perform a node restart. Interconnect changes for the database occur at instance startup. However, the interconnect for Oracle Clusterware might be different.
- Because interconnects are chosen when instances start, just issuing OIFCFG commands does not have an immediate effect on the running system. Instead, changes take effect after restarting the component that might be affected by the command.

OIFCFG Examples

The following examples show some common uses for the OIFCFG commands.

Example 1 Listing the Names of Network Interfaces

You can use OIFCFG to list the interface names and the subnets of all of the interfaces available on the local node by executing the `iflist` keyword, as shown in this example:

```
oifcfg iflist
hme0     139.185.141.0
qfe0     204.152.65.0
```

Example 2 Retrieving Network Information

You can also retrieve specific OIFCFG information with a `getif` command.

```
oifcfg getif [ [-global | -node nodename] [-if if_name[/subnet]] [-type if_type] ]
```

For example, after you install Oracle Clusterware, you can verify that the public and cluster interconnect have been set to the desired values by entering the following commands as `root`:

```
$ oifcfg getif
```

This command should return values for global public and global cluster_interconnect. For example:

```
en0 144.25.68.0 global public
hme0 139.185.141.0 global cluster_interconnect
```

Example 3 Storing a New Global Interface

To store a new interface, use the `setif` keyword. For example, to store the interface `hme0`, with the subnet 139.185.141.0, as a global interface (to be used as an interconnect for all of the Oracle RAC instances in your cluster), you would use the command:

```
oifcfg setif -global hme0/139.185.141.0:cluster_interconnect
```

For a cluster interconnect that exists between only two nodes, for example `rac1` and `rac2`, you could create the `cms0` interface with the following commands, assuming 139.185.142.0 is the subnet number for the interconnect on `rac1` and `rac2`, respectively:

```
oifcfg setif -global cms0/139.185.142.0:cluster_interconnect
```

Example 4 Deleting the Stored Interface

Use the OIFCFG `delif` command to delete the stored configuration for global or node-specific interfaces. A specific node-specific or global interface can be deleted by supplying the interface name, with an optional subnet, on the command line. Without the `-node` or `-global` options, the `delif` keyword deletes either the given interface or all of the global and node-specific interfaces on all of the nodes in the cluster.

For example, the following command deletes the global interface named `qfe0` for the subnet 204.152.65.0:

```
oifcfg delif -global qfe0/204.152.65.0
```

The following command deletes all of the global interfaces assigned with OIFCFG:

```
oifcfg delif -global
```

D

High Availability Oracle Clusterware Command-Line Reference and C API

This appendix describes the Oracle Clusterware application program interface (API) command reference and includes the following topics:

- Using Oracle Clusterware Commands
- Oracle Clusterware Commands
- C Application Programming Interface to Oracle Clusterware
- Functions for Managing Resource Structures

> **See Also:** Chapter 5, "Making Applications Highly Available Using Oracle Clusterware" for detailed information about using Oracle Clusterware to make applications highly available

Using Oracle Clusterware Commands

This section explains how to use the Oracle Clusterware commands to manage applications and application resources under the Oracle Clusterware framework.

Application Profile Syntax

The examples in this appendix show the command syntaxes that you use in application profiles. Lines starting with a pound sign (#) are comment lines and are not processed as part of a profile. A backslash (\) at the end of a line indicates that the next line is a continuation of the previous line. See the section titled "Using the crs_profile Command to Create An Application Resource Profile" on page 5-8 to see a profile example.

Security and Permissions

Oracle Clusterware uses security similar to Linux or UNIX systems, where permissions may be specified for the owner, nodes of a group, or holders of specific privileges. There may also be default permissions for other users. These permissions are `read`, `write`, and `run` on Windows systems, and `rwx` on Linux or UNIX systems. Default ownership and permissions are set when you create an application profile. Table D–1 shows the Oracle Clusterware commands and their owners and describes which commands require write permission and which commands require run permission.

Table D–1 Oracle Clusterware Command Owner and Permissions Matrix

Owner	Read	Write Permission Required	Run/Execute Permission Required

User	`crs_stat {any options}`	`crs_register {no arguments}`
		`crs_register -u`
		`crs_unregister`
			`crs_start`
			`crs_stop`
			`crs_relocate`
			`crs_stat` {no arguments}
			`crs_stat -l -t -r`

Group	`crs_stat {any options}`	`crs_register {no arguments}`
		`crs_register -u`
		`crs_unregister`
			`crs_start`
			`crs_stop`
			`crs_relocate`
			`crs_stat` {no arguments}
			`crs_stat -l -t -r`

Other	`crs_stat {any options}`	`crs_register {no arguments}`
		`crs_register -u`
		`crs_unregister`
			`crs_start`
			`crs_stop`
			`crs_relocate`
			`crs_stat` {no arguments}
			`crs_stat -l -t -r`

Note: On platforms that do not have a group concept, the group permissions are `NULL` or not set.

Note: **Do not use** the Oracle Clusterware commands prefixed with `crs_` (except for `crs_stat`) on resources that have names beginning with the prefix `ora` unless either Oracle Support asks you to, or unless Oracle has certified you as described in `https://metalink.oracle.com`. Server Control (SRVCTL) is the correct utility to use on Oracle resources. You can create resources that depend on resources that Oracle has defined. You can also use the Oracle Clusterware commands to inspect the configuration and status.

Oracle Clusterware Commands

Table D–2 alphabetically lists the Oracle Clusterware commands that this appendix describes. Note that the `-q` option is available for all commands and enables the command to run in quiet mode. This means that no messages are displayed on the console.

Table D–2 Oracle Clusterware Commands Summary

Command	Description
crs_getperm on page D-3	Lists the permissions associated with a resource.
crs_profile on page D-4	Creates, validates, deletes, and updates an Oracle Clusterware application profile

Table D–2 (Cont.) Oracle Clusterware Commands Summary

Command	Description
crs_register on page D-16	Registers configuration information for an application with the OCR.
crs_relocate on page D-9	Relocates an application profile to another node.
crs_setperm on page D-12	Sets permissions associated with a resource.
crs_stat on page D-12	Lists the status of an application profile.
crs_start on page D-14	Starts applications that have been registered.
crs_stop on page D-15	Stops an Oracle Clusterware application.
crs_unregister on page D-16	Removes the configuration information for an application profile from the OCR.

Note: On platforms that do not have a group concept, the group permissions are `NULL` or not set.

crs_getperm

Lists the permissions defined for the resource. If any user other than `root` needs access to a resource that was created by the `root` user, you need to explicitly give them access. However, resources registered by users other than `root` are owned by the registering user, and can see their own resources. The access definitions are similar to Unix.

See Also: The crs_setperm command on page D-11 to modify the permissions associated with a resource

Syntax and Options for crs_getperm

Use the `crs_getperm` command with the following syntax:

```
crs_getperm resource_name [-u user|-g group] [-q]
```

To obtain the permissions associated with a resource, use the following syntax:

```
crs_getperm resource_name
```

Example of crs_getperm

To list the permissions associated with the `postman` application, use the following command:

```
crs_getperm postman

====================
Name: postman
owner:root:rwx,pgrp:oinstall:r-x,other::r--,user:oracle:r-x,
====================
```

Table D–3 crs_getperm Options

Option	Description
`-g` *group*	Obtains the permissions associated with the specified (operating system) group.

Table D–3 (Cont.) crs_getperm Options

Option	Description
-q	Runs the command in quiet mode (no messages are displayed on the console).
resource_name	Obtains the permissions associated with a specific resource.
-u *user*	Obtains the permissions associated with the specified user.

crs_profile

Creates, validates, deletes, and updates an Oracle Clusterware application profile. It works on the user's copy of a profile.

You can also use crs_profile to generate a template script. The crs_profile command creates new application profiles and validates, updates, deletes, or lists existing profiles. An application profile assigns values to attributes that define how a resource should be managed or monitored in a cluster. For the root user, profiles are written to the *CRS_home*/crs/profile directory. For nonprivileged users, the profile is written to the *CRS_home*/crs/public directory. Values in the profile that are left blank are ignored and may be omitted if not required. Omitted profile variables that are required for a resource type can cause validation or registration to fail.

After you have created an application profile and registered the application with Oracle Clusterware using the crs_register command, you can use other Oracle Clusterware commands, such as crs_stat, crs_start, crs_stop, crs_relocate, and crs_unregister, on the application. You must register applications using the crs_register command *before* the other Oracle Clusterware commands are available to manage the application. The crs_profile command may have other options for defining user-defined attributes in a profile.

Syntax and Options for the crs_profile Command

Use the crs_profile command with the following syntax to create an application profile template:

```
crs_profile -create resource_name -t application [-a action_script] [-B
 executable_pathname] [-dir directory] [-d description] [-p placement_policy] [-h
 hosting_nodes] [-r required_resources] [-l optional_resources]  [-o option,[...]]
 [attribute_flag attribute_value] [...] [-f] [-q]

crs_profile -create resource_name -t application
          [-dir directory_path] [-a action_script] [-B binary_pathname]
          [-d description] [-h hosting_members] [-r required_resources]
          [-l optional_resources] [-p placement_policy]
          [-o as=auto_start,ci=check_interval,ft=failure_threshold,
          fi=failure_interval,ra=restart_attempts,fd=failover_delay,
          st=script_timeout,ap=active_placement,bt=rebalance,
          ut=uptime_threshold,rt=start_timeout,pt=stop_timeout] [-f] [-q]
```

To create an application profile from an application profile template you use crs_profile to create a template (a cap) file. This file is then read automatically by the crs_register command which reads the data from the cap file and puts it in the OCR. For example:

```
crs_profile -create resource_name -I template_file [-f] [-q]
```

To validate the application profile syntax of a profile, enter the following command:

```
crs_profile -validate resource_name [-q]
```

To list one or more application profiles:

```
crs_profile -print [resource_name [...]] [-q]
```

To create an application profile template from an existing application profile:

```
crs_profile -template resource_name [-O template_file] [-q]
```

To update an application profile:

```
crs_profile -update resource_name [option [...]] [-q]
```

To delete an application profile and its associated action program:

```
crs_profile -delete resource_name [-q]
```

Note: The `crs_profile -delete` command deletes the resource profile file but does not delete the action script file.

Table D–4 Options for the crs_profile Command

Option	Description

[*attribute_flag attribute_value*]	All `crs_profile` commands take the following format:
	`crs_profile attribute_flag attribute_value`
	This command sets the specified attribute (whose flag is specified by *attribute_flag*) to the given value (whose value is specified by *attribute_value*) in the profile that gets generated. For example, the following command sets the `-validate` attribute to the value `resource_name`:
	`crs_profile -validate resource_name`

[`-a` *action_script*]	Specifies the action program for the application. An action program has start, stop, and check entry points that are called by Oracle Clusterware. You can specify either a full path name for the script file or its filename.
[`-B` *executable_pathname*]	Specifies the location of the application executable and causes the `crs_profile` command to generate an action program file that contains the application executable path name.
`-create` *resource_name*	Creates a resource profile for the indicated application according to the specified options. A *resource_name* is a string containing a combination of characters [`a-z, A-Z, 0-9,'.','_'`]. The resource name may not start with a period (`.`). The maximum length for a resource name is 128 characters. Resource profiles are created in the directory that you specify.
[`-d` *description*]	Specifies the description of the application. Enclose the description in double quotation marks (`""`) if it contains white space. If you do not specify an application description, then Oracle Clusterware uses the application name.
[`-dir` *directory*]	Specifies the file location where the profile is located.
[`-h` *hosting_nodes*]	Specifies an ordered list of nodes, separated by white space, that can host the application. If there is more than one node, then the list must be enclosed in double quotation marks (`""`). If you specify the `-p` option with a placement policy of `preferred` or `unavailable`, then you must also specify the `-h` option.
[`-l` *optional_resources*]	Specifies an ordered list of optional resources, separated by white space. If there are multiple optional resources specified, then the list must be enclosed in double quotation marks (`""`).
[`-o` *option*[,...]]	Specifies a comma-delimited list of attribute option names and their values. To see the list of attribute option names, see Table D–5, " Options for the -o Flag" on page D-6 or run the `crs_profile -help` command.

Table D–4 (Cont.) Options for the crs_profile Command

Option	Description
[-p *placement_policy*]	Specifies the policy according to which Oracle Clusterware selects the node on which to start or restart the application. You can specify `balanced`, `favored`, or `restricted` as placement policies. All policies except for `balanced` require you to also specify the `-h` flag.
`-q`	Runs the command in quiet mode (no messages are displayed on the console).
[`-r` *required_resources*]	Specifies an ordered list of resources, separated by spaces, on which the application depends. These resources must be active on any node on which the application is running. If there are multiple required resources, then the list must be enclosed in double quotation marks (`""`). If you do not specify a required resources list, then Oracle Clusterware imposes no required dependencies upon the application resource.
`-t` *application*	Indicates an application or application resource type.
`-validate`	Validates the application profile syntax of a profile.

Table D–5 Options for the -o Flag

Option	Description
`ap=active_placement`	When set to `1`, it reevaluates the placement policy when a new cluster node becomes available. With placement policies of `FAVORED` and `RESTRICTED`, a highly available application relocates to a more highly favored cluster node. The default is `0`.
`as=auto_start`	When set to `1`, the application automatically starts after a cluster restart, regardless of whether it had been stopped or running before the restart. When set to 0,automatically starts the application only if it had been running before the restart. The default is `0`.
`ci=check_interval`	Sets the time (in seconds) at which the check entry point of the application's action program runs. The check interval is the maximum amount of time that an application can be unavailable to clients before Oracle Clusterware attempts to restart it. The default check interval is 60 seconds.
`fd=failover_delay`	Specifies the number of seconds that Oracle Clusterware waits before attempting to relocate the application resource. An application resource that was running on a cluster node that failed restarts immediately on that node if it becomes available again within the failover delay period. If application resources are dependent on each other due to required resources defined in application profiles, then all interdependent application resources wait for the largest failover delay that you have defined for any of the resources. The default failover delay is `0` (zero) seconds.
`fi=failure_interval`	Specify the time (in seconds) during which the failure threshold is calculated and applied. The default failure interval is `0` (zero). Oracle Clusterware uses a default failure interval of `0`, which turns off monitoring of failure threshold and failure intervals. Specifying a nonzero failure interval has no effect unless the failure threshold is also nonzero.
`ft=failure_threshold`	Specifies the number of times that Oracle Clusterware may detect an application or application resource failure within the failure interval before it marks the application or application resource as unavailable and stops monitoring it. The value must be in the range `0-20`. Setting the value to `0` (zero) turns off failure threshold monitoring. If you do not specify a failure threshold, then Oracle Clusterware uses a default failure threshold of `0` (zero).
`ra=restart_attempts`	Specifies the number of times that Oracle Clusterware attempts to restart the application or application resource on the current node before attempting to relocate it. The default number of restart attempts is `1`.
`st=script_timeout`	Sets the maximum number of seconds within which an action program can run. The Oracle Clusterware commands and daemon return an error message and post an event to Event Manager (EVM) when they invoke action program entry points and the script does not complete its execution within this time. The default is 60 seconds.

Examples of the crs_profile Command

The following is an example of using the `crs_profile` command to create the application profile for the `postman` application:

```
crs_profile -create postman \
-t application \
-a postman.sh \
-o ci=5,ra=2"
```

The following is an example of using the `crs_profile` command to validate the application profile named `dtcalc`:

```
crs_profile -validate dtcalc
```

If you do not specify either the `-a` or `-B` options, then the profile is created without an action program value. You should create a readable, executable script and update the profile with this script's value before attempting to start the application. If you specify the `-a` option alone, then the action program specified must exist at the specified location or in the default directory if no path is specified. Otherwise, the command fails.

crs_register

The `crs_register` command registers one or more applications specified with the *`resource_name`* parameter for each application. This command requires write access to the target application. This command only succeeds if the profile is found either in the default location or in the directory specified by the `-dir` option. An application must be registered for Oracle Clusterware to monitor the resource or to start, restart, or relocate a highly available application associated with an application resource. An application registration must be updated for any changes to an application profile to take effect. Also, see the "Security and Permissions" section on page D-1 for information about the privileges required for this command.

An application can be registered or updated only if the CRS daemon is active and an Oracle Clusterware application profile exists in the profile directory for this application. If fields are missing from an application profile, then the profile is merged with the default profile template and Oracle uses the values in the default profile template.

The ownership and default permissions for the application are set during the registration process. You can register any `.cap` file as long as the permissions for the file permit `read` and `write`, for example, `crs_profile` and `crs_register` must be able to read the file. By default, the user who registers the application is the owner of the application. If the profile cannot be registered, then Oracle Clusterware displays messages explaining why. Use the `crs_stat` command to verify that the application is registered.

Syntax and Options for the crs_register Command

You can use the `crs_register` command to register and update applications. Use the following `crs_register` syntax to register an application:

```
crs_register resource_name [-dir directory_path] [...] [-u] [-f] [-q]

crs_register resource_name -update [-a action_script]
[-d description] [-h hosting_members] [-r required_resources]
[-l optional_resources] [-p placement_policy]
[-o as=auto_start,ci=check_interval,ft=failure_threshold,
fi=failure_interval,ra=restart_attempts,fd=failover_delay,
```

```
st=script_timeout,ap=active_placement,bt=rebalance,
ut=uptime_threshold,rt=start_timeout,pt=stop_timeout] [-q]
```

The *resource_name* [...] parameter can be the name of one or more application resources as specified in an application profile. If you do not specify any options, then the crs_relocate command relocates each specified application resource according to its placement policy and required resource lists. Oracle Clusterware does not relocate a resource if there are interdependent application resources unless you specify the -f option. A profile must exist for the application that you are registering.

Table D–6 Options on the crs_register Command

Option	Description
[-dir *directory_path*]	Specifies the directory where the .cap file is located if the .cap file is not located in the default directory.
-f	Enables you to register a resource even though you have no execute permission on one or more required resources.
-u	Ensures the changes take effect immediately. Use crs_register -u immediately after issuing a crs_profile -update command or after manually editing an application profile.
[*attribute_flag attribute_value*]	All crs_profile commands take the following format: crs_profile *attribute_flag attribute_value* This command sets the specified attribute (whose flag is specified by *attribute_flag*) to the given value (whose value is specified by *attribute_value*) in the profile that gets generated. For example, the following command sets the -validate attribute to the value resource_name: crs_profile -validate resource_name
[-a *action_script*]	Specifies the action program for the application. An action program has start, stop, and check entry points that are called by Oracle Clusterware. You can specify either a full path name for the script file or its filename.
[-d *description*]	Specifies the description of the application. Enclose the description in double quotation marks ("") if it contains white space. If you do not specify an application description, then Oracle Clusterware uses the application name.
[-h *hosting_nodes*]	Specifies an ordered list of nodes, separated by white space, that can host the application. If there is more than one node, then the list must be enclosed in double quotation marks (""). If you specify the -p option with a placement policy of preferred or unavailable, then you must also specify the -h option.
[-l *optional_resources*]	Specifies an ordered list of optional resources, separated by white space. If there are multiple optional resources specified, then the list must be enclosed in double quotation marks ("").
[-o *option*[,...]]	Specifies a comma-delimited list of attribute option names and their values. To see the list of attribute option names, see Table D–5, " Options for the -o Flag" on page D-6 or run the crs_profile -help command.
[-p *placement_policy*]	Specifies the policy according to which Oracle Clusterware selects the node on which to start or restart the application. You can specify balanced, favored, or restricted as placement policies. All policies except for balanced require you to also specify the -h flag.
-q	Runs the command in quiet mode (no messages are displayed on the console).
[-r *required_resources*]	Specifies an ordered list of resources, separated by spaces, on which the application depends. These resources must be active on any node on which the application is running. If there are multiple required resources, then the list must be enclosed in double quotation marks (""). If you do not specify a required resources list, then Oracle Clusterware imposes no required dependencies upon the application resource.

Use the following crs_register command syntax to update a registered application:

```
crs_register resource_name -update [option ...] [-o option,...] [-q]
```

> **See Also:** Table D–4 and Table D–5 for the listing of options and their definitions for using -update; the options for -update are the same as for the crs_profile command

Example of the crs_register Command

The following example registers an application named postman for which an application profile exists in the *CRS_home*/crs/profile directory:

```
CRS_home/bin/crs_register postman
```

> **Note:** The profile will be in the crs/profile directory if the profile is created by the root user. The profile will be in the crs/public directory if the profile is created by any other user.

Related Commands for the crs_register Command

crs_stat, crs_profile, crs_relocate, crs_start, crs_stop, crs_unregister

crs_relocate

The crs_relocate command relocates applications and application resources as specified by the command options that you use and the entries in your application profile. The specified application or application resource must be registered and running under Oracle Clusterware in the cluster environment before you can relocate it. The command displays a message if you specify a cluster node that is unavailable or if the attempt to relocate failed. Also, see the "Security and Permissions" section on page D-1 for information about the privileges required for this command.

When you issue a crs_relocate command, Oracle Clusterware first runs the stop entry point of the action program on the node on which it is currently running. Oracle Clusterware then performs the start entry point of the action program to start it on a new node.

If Oracle Clusterware fails to stop the application or application resource on the current node due to an action program error, then it marks it as UNKNOWN. You cannot issue the crs_relocate command on a resource in this state. Instead, run a crs_stop -f command on the resource and restart it by issuing he crs_start command to return it to the ONLINE state before you attempt to relocate it again. If Oracle Clusterware fails to restart an application resource, then you may need to check the resource action program.

If the action program start entry point fails to run successfully, then the stop entry point is run. If the stop entry point fails to run successfully, then the state is marked as UNKNOWN and relocation attempts are stopped. If the stop entry point succeeds, then the state is set to OFFLINE. The target state remains ONLINE however, so subsequent cluster node failures or restarts can cause Oracle Clusterware to attempt to restart the application. If you have not specified the node to which to relocate it and if there are available cluster nodes that satisfy the placement criteria, then Oracle Clusterware attempts to start the application on one of these available nodes.

If one or more user-defined attributes have been defined for application resources, then you can specify values for these attributes when relocating an application with the `crs_relocate` command. The specified value is passed to the action program as an environment variable with the attribute name.

The actions that Oracle Clusterware takes while relocating an application resource are echoed on the command line. You can also monitor them using the Event Manager (EVM). Standard error and standard output from a resource action program that is started by the `crs_relocate` command are redirected to the standard error and standard output for the `crs_relocate` command. Note that if the Oracle Clusterware daemon starts an application, then standard error and standard output of the action program is lost. In an action program, you can check for user invocation of the action program using reason codes.

Syntax and Options for the crs_relocate Command

Use the `crs_relocate` command with the following syntax:

```
crs_relocate resource_name [...] [-c cluster_node] [-f] [-q]

crs_relocate resource_name [-c cluster_node] [-q]

crs_relocate [USR_attribute_name=value] [...] resource_name [-c cluster_node] [-q]

crs_relocate -s source_node [-c cluster_node] [-q]
```

Table D–7 Options on the crs_relocate Command

Option	Description
`-c` *`cluster_node`*	Relocates each indicated application or application resource to the specified node regardless of its placement policy. If required resources are not available on the destination node or if the application resource is restricted from that node, then the `crs_relocate` command fails and the application or application resource remains on the current node.
`-f`	Forces relocation of the specified applications, all applications dependent on them and all applications that they are dependent upon. This option is necessary for relocating any application that requires another application or one that is required by any `ONLINE` application.
`-s` *`source_node`*	Relocates all running applications or application resources from the *`source_node`*. If you do not also specify the `-c` option, then the `crs_relocate` command relocates each resource according to its placement policy and required resource lists.
`-t`	Lists information for all resources in a tabular form.
`-q`	Runs the command in quiet mode (no messages are displayed on the console).
[`USR_attribute_name=`*`value`*]	When starting the resource, this option sets the named attribute to the given value in the CRS resource's environment.

Example of the crs_relocate Command

The following example relocates an application resource to the node known as `rac1`:

```
crs_relocate postman -c rac1
Attempting to stop `postman` on node `rac2`
Stop of `postman` on node `rac1` succeeded
Attempting to start `postman` on node `rac1`
Start of `postman` on node `rac1` succeeded
```

The following example attempts to relocate all application resources from node rac2 to node rac1:

```
crs_relocate -s rac2 -c rac1
Attempting to stop `postman` on node `rac2`
Stop of `postman` on node `rac2` succeeded.
Attempting to start `postman` on node `rac1`
Start of `postman` on node `rac1` succeeded.
Attempting to stop `calc` on node `rac2`
Stop of `calc` on node `rac2` succeeded.
Attempting to start `calc` on node `rac1`
Start of `calc` on node `rac1` succeeded.
```

If a user-defined attribute USR_DEBUG has been defined, then the following example runs the stop and start entry point of the action program with the USR_DEBUG environment variable set to FALSE. This overrides any value set in the application profile. In the corresponding action program, if you add the following line to the appropriate section of the action program, then you can view the value:

```
echo $USR_DEBUG
```

Then issue the following command:

```
# crs_relocate USR_DEBUG=false database
```

crs_setperm

Modifies the permissions associated with a resource. This command is similar to the chmod command for Linux and UNIX systems or the Windows desktop options, in this order: File, Properties, Security, and Permissions.

> **See Also:** The crs_getperm command on page D-3 to obtain the permissions associated with a resource

Syntax and Options for the crs_setperm Command

Use the crs_setperm command with the following syntax:

```
crs_setperm resource_name -u aclstring [-q]

crs_setperm resource_name -x aclstring [-q]

crs_setperm resource_name -o user_name [-q]

crs_setperm resource_name -g group_name [-q]
```

Table D–8 Options on the crs_setperm Command

Option	Description
-u	Updates the ACL string.
-x	Deletes the ACL string.
-o	Changes the owner of the resource. **Note:** Only the root user can change the owner.
-g	Changes the primary group of the resource.
-q	Runs the command in quiet mode (no messages are displayed on the console).

Table D–8 (Cont.) Options on the crs_setperm Command

Option	Description

aclstring	Is one of the following:
	`user:`*username*`:rwx`
	`group:`*groupname*`:r-x`
	`other::r--`

Example of the crs_setperm Command

The following example modifies the permissions on the `admin1` user for the `postman` application:

```
crs_setperm postman -u user:admin1:r-x
```

crs_stat

The `crs_stat` command provides status information for resources on the cluster nodes. To query resources with the `crs_stat` command, the CRS resource permissions must have read and execute permissions (`r` and `x` permissions on Linux and UNIX systems). An exception is the `-g` option, that anyone can use to verify whether a resource exists.

Resources are either `ONLINE` or `OFFLINE` as shown in the `STATE` attribute. An application resource in the `ONLINE` state is running successfully on a cluster node. This cluster node is shown indicating its state.

The `TARGET` value shows the state to which Oracle Clusterware attempts to set the resource. If the `TARGET` value is `ONLINE` and a cluster node fails, then Oracle Clusterware attempts to restart the application on another node if possible. If there is a condition forcing a resource `STATE` to be `OFFLINE`, such as a required resource that is `OFFLINE`, then the `TARGET` value remains `ONLINE` and Oracle Clusterware attempts to start the application or application resource once the condition is corrected.

A `TARGET` value for all non-application resources should be `ONLINE` unless the resource has a failure count higher than the failure threshold, in which case the `TARGET` is changed to `OFFLINE`. Oracle Clusterware then treats the resource as if its `STATE` were `OFFLINE`. If the `STATE` is `ONLINE` and the `TARGET` is `OFFLINE`, then you can reset the target value to `ONLINE` using the `crs_start` command.

The verbose status `-v` gives additional information that may be useful, especially for troubleshooting. The `RESTART_COUNT` value shows how many times an application resource has been restarted on a single cluster node. The maximum number of restarts before Oracle Clusterware stops restarting the application is equal to `RESTART_ATTEMPTS`. `FAILURE_COUNT` shows the number of times that a resource has failed within the `FAILURE_INTERVAL` defined in the application profile. The maximum number of failures before Oracle Clusterware stops attempting to restart an application is equal to the value set for the `FAILURE_THRESHOLD` parameter. If a cluster node fails and applications are waiting to be relocated due to the profile `FAILOVER_DELAY` attribute, then the verbose status also shows the `FAILOVER_STATUS` field. `The FAILOVER_STATUS` field is not shown at any other time. The `FAILOVER_STATUS` field shows the node the application failed on and how much time is left waiting for that node to restart before restarting on another node.

Syntax and Options for the crs_stat Command

Use the `crs_stat` command with the following syntax:

```
crs_stat [resource_name [...]] [-v] [-l] [-q] [-c cluster_node]
```

```
crs_stat [resource_name [...]] -t [-v] [-q] [-c cluster_node]

crs_stat -p [resource_name [...]] [-q]

crs_stat [-a] resource_name -g

crs_stat [-a] resource_name -r [-c cluster_node]

crs_stat -f [resource_name [...]] [-q] [-c cluster_node]

crs_stat -ls resource_name
```

To view information about all resources, enter the command with no arguments.

Table D–9 Options on the crs_stat Command

Option	Description
-a *resource_name*	Used with the -g or -r option to verify whether the specified resource is registered or running under Oracle Clusterware. Specify the name of the resource as listed in its application profile.
-c *cluster_node*	Displays information about applications or application resources on the specified cluster node.
-f	Displays all information about the resource including extended information (-v) and the in-memory profile (-p).
-g	Returns 0 (zero) if the specified application or application resource is registered with Oracle Clusterware; returns 1 if it is not. This option only works if a single resource is specified with the -a option.
-l	Displays the status in a list (nontabular) format. This is the default format.
-ls	Lists resources, owners, and permissions for all resources.
-ls *resource_name*	Lists resources, owners, and permissions for a specified resource.
-p *resource_name* [...]]	Displays the status of the in-memory profile for the specified resources. If no resources are specified, then the status of the in-memory profile for all registered resources is displayed. If a resource is not registered, its status is not displayed.
-q	Runs the command in quiet mode (no messages are displayed on the console).
-r	Returns 0 (zero) if the specified application or application resource is running under Oracle Clusterware; returns 1 if it is not. This option can only be successful if a single resource is specified with the -a option.
-t	Lists information for all resources in a tabular form.
-v	Lists how many times a resource has been restarted or has failed within the resource failure interval, the maximum number of times that a resource can be restarted or can fail, and the target state of the application. Also, this option lists normal status information. This option displays extended information about the status of resources. Extra attributes shown include RESTART_COUNT and FAILURE_COUNT. The RESTART_COUNT attribute shows the number of restarts of the application that have been attempted since it was started. The FAILURE_COUNT attribute shows how many failures have occurred during the last FAILURE_THRESHOLD in seconds. The attribute FAILOVER_STATUS shows the time at which an application resource started while waiting to relocate due to a cluster node failure if the resource has a FAILOVER_DELAY value greater than 0. It is not displayed otherwise.

Examples of the crs_stat Command

The following example displays the status information for the postman application:

```
crs_stat postman
```

```
NAME=postman
TYPE=application
TARGET=ONLINE
STATE=ONLINE on rac2
```

The following example displays the status information in a tabular form:

```
crs_stat -t

Name         Type        Target      State     Host
------------------------------------------------------------
cluster_lockd application ONLINE    ONLINE     rac2
dhcp          application OFFLINE   OFFLINE
```

The following example shows the output for the `crs_stat` command with the `-v` option for the node `rac2`:

```
crs_stat -v

NAME=cluster_lockd
TYPE=application
RESTART_ATTEMPTS=30
RESTART_COUNT=0
FAILURE_THRESHOLD=0
FAILURE_COUNT=0
TARGET=ONLINE
STATE=ONLINE on rac2
```

crs_start

The `crs_start` command sets an application or application resource target state to `ONLINE` and attempts to start specified registered applications or application resources. The target state is the state that Oracle Clusterware attempts to achieve. Also, see the "Security and Permissions" section on page D-1 for information about the privileges required for this command.

There are several user-defined attributes available for starting an application with `crs_start`. The specified value is passed to the action program as an environment variable with the attribute name.

If you do not specify one node on which to start the application and there are available cluster nodes that satisfy the placement criteria, then Oracle Clusterware attempts to start the application on one of the available nodes. You must stop a resource that has failed before restarting it. You should also confirm why your action program is failing.

The `crs_start` command starts dependent resources as defined in the application profile `REQUIRED_RESOURCE` fields. After a registered resource is started, you must use the `crs_stop` command to end its execution.

The `crs_start` command displays status messages and if there are problems starting an application, then feedback is displayed. Standard error and standard output from a resource action program invoked by the `crs_start` command are redirected to the standard error and standard output for `crs_start`. Note that if the CRS daemon starts an application, then standard error and standard output of the action program is lost. Within an action program, you can check for user invocation of the action program using reason codes.

Syntax and Options for the crs_start Command

Use the `crs_start` command with the following syntax:

```
crs_start resource_name  [...] [-c cluster_node] [-q] [-f]

crs_start -all [-q]

crs_start [USR_attribute_name=value] [...] resource_name [-c node_name] [-q]
```

For *resource_name* [...], the names are as specified in an application profile of one or more resources to be started. The resource must be registered with Oracle Clusterware.

Table D–10 Options On the crs_start Command

Option	Description
-all	Starts all registered Oracle Clusterware applications or application resources on active cluster nodes, according to their placement policies and required resource lists.
-c *node_name*	Starts each indicated resource on the specified node if the cluster node is enabled by the placement policy and resource dependencies. If the cluster node specified is not enabled by the placement policy and resource dependencies, then the crs_start command fails. If the specified node is not available, the command fails. If one of the specified resources fails to start, then the command still attempts to start other resources listed on the command line.
-f	Forces starting of a specified application or application resource if all of the required resources are available or can be started. Any applications or application resources in the specified application profile's required resource list are started if not currently running or are relocated if they are running on another cluster node. Use this option with only one specified application resource at a time.
-q	Runs the command in quiet mode (no messages are displayed on the console).
[USR_attribute_name=*value*]	

Examples of the crs_start Command

The following example starts an application named postman:

```
crs_start postman
Attempting to start `postman` on node `rac1`
Start of `postman` on node `rac1` succeeded.
```

The following example starts the application on a specific cluster node rac2:

```
crs_start -c rac2 postman
Attempting to start `postman` on node `rac2`
Start of `postman` on node `rac2` succeeded.
```

crs_stop

Stops an application on the specified node. You can run the crs_stat command to verify that the application you specify is stopped.

Syntax and Options for the crs_stop Command

Use the crs_stop command with the following syntax:

```
crs_stop resource_name [...] [-f] [-q]

crs_stop -c cluster_node [...] [-q]

crs_stop -all [-q]

crs_stop [USR_attribute_name=value] [...] resource_name [-q]
```

`-c cluster_node [...]`

Specifies one or more active cluster nodes on which to stop all of the applications or application resources. You can use the options described in Table D–11:

Table D–11 Options on the crs_stop Command

Option	Description
`-all`	Stops all registered Oracle Clusterware applications or application resources on active cluster nodes.
`-c` *`cluster_node`*	Stops applications or application resources on the specified cluster node.
`-f`	Forces termination of the indicated applications or application resources and all application resources dependent on the specified resources. This option is useful to stop all application resources when the `crs_stop` command fails because of application resources that are dependent on the specified resource or if the application is in the `UNKNOWN` state because of an unsuccessful stop. This option makes the `crs_stop` command ignore any errors that are reported by an action program.
-q	Runs the command in quiet mode (no messages are displayed on the console).

Examples of the crs_stop Command

To stop an application named `dtcalc`, enter the following command:

```
crs_stop dtcalc
```

Status Messages for the crs_stop Command

Oracle Clusterware displays the following status messages:

```
Attempting to stop `dtcalc` on node `rac2`
Stop of `dtcalc` on node `rac2` succeeded.
```

crs_unregister

The `crs_unregister` command removes the registration information of Oracle Clusterware resources from the binary Oracle Clusterware registry database. Oracle Clusterware will no longer acknowledge this resource. An application associated with a resource that is unregistered is no longer highly available. Also, see the "Security and Permissions" section on page D-1 for information about the privileges required for this command.

Upon successful completion of the `crs_unregister` command, the resource is removed from the online Oracle Clusterware environment. You cannot unregister a resource that is a required resource for another resource. You must stop the resource by using the `crs_stop` command before unregistering it.

Syntax and Options for the crs_unregister Command

Use the `crs_unregister` command with the following syntax:

```
crs_unregister resource_name [...] [-f] [-q]
```

You can use the options described in Table D–12:

Table D–12 Options on the crs_unregister Command

Option	Description
-f	Forces removal of the indicated applications or application resources and all application resources dependent on the.indicated resource. This option is useful to unregister all application resources when the crs_unregister command fails because of resources that are dependent on the specified resource or if the application is in the UNKNOWN state because of an unsuccessful stop.
-q	Runs the command in quiet mode (no messages are displayed on the console).

Example of the crs_unregister Command

The following example unregisters a highly available application called postman:

```
crs_unregister postman
```

The following example forcefully unregisters the listener:

```
crs_unregister ora.LISTENER_node-name.lsnr -f
```

C Application Programming Interface to Oracle Clusterware

The APIs described in this section provide access to operational control of resources that Oracle Clusterware manages. The API is used to register user applications to the Oracle Clusterware system so that they can be managed by Oracle Clusterware and made highly available. When an application is registered, the application can be started and its state queried. If the application is no longer to be run, it can be stopped and unregistered from Oracle Clusterware. The Oracle Clusterware services are provided by another process, such as the CRSD process, running outside the database server. These APIs communicate with the CRSD process using an IPC mechanism.

Note: You can install the Oracle Clusterware high availability Application Programming Interface (API) from the Oracle Database client installation media.

clscrs_init_crs

Initializes a context for communications with Oracle Clusterware.

Parameters

- ctx—Context to initialize, caller allocated storage
- errf—A Callback function for the error info, or NULL
- errCtx—Context for the error callback func, or NULL
- flags—CLSCRS_FLAG_TRACE, to enable tracing of lower layers
- CLSCRS_FLAG_USETHREADS—Enables using the clscrs context in multiple threads

Returns

Status enum.

Syntax

```
CLSCRS_STAT clscrs_init_crs(clscrs_ctx **ctx, clscrs_msgf errf, void *errCtx, ub4
```

```
flags);
```

clscrs_term_crs

Releases a context for communications with Oracle Clusterware.

Parameter

ctx—Context previously initialized with clscrs_init_crs.

Returns

status enum

Syntax

```
CLSCRS_STAT clscrs_term_crs( clscrs_ctx **ctx);
```

clscrs_getnodename

Returns the correct node name. If you issue this command within a configured cluster, then this is the name as known to the clusterware. If not in a cluster, then the name is as known through other clusterware means. Getting a node name may require a lot of resources in the first call, especially if CLSCRS was not initialized with a preexisting CSS context.

Parameters

- ctx—Context previously initialized with clscrs_init
- node_name—Filled in node name

Syntax

```
CLSCRS_STAT clscrs_getnodename(clscrs_ctx *ctx, oratext *node_name);
```

clscrs_env_create

Creates a call environment describing the parameters to the Oracle Clusterware calls. The memory is owned by clscrs which is released by a call to clscrs_env_delete.

Parameters

- context—Input context
- env—Output environment pointer

Returns

status enum

Syntax

```
CLSCRS_STAT clscrs_env_create( clscrs_ctx *ctx, clscrs_env *env );
```

clscrs_env_set

Sets an attribute or value argument in an environment for the Oracle Clusterware calls; the value may be NULL.

Parameters

- env—Previously created environment
- attr—Attribute name
- val—Attribute value, may be NULL pointer

Returns

environment status enum

Syntax

```
CLSCRS_ENVCODE clscrs_env_set(clscrs_env env, const  oratext *attr, const oratext
*val);
```

clscrs_env_delete

Deletes an environment for the Oracle Clusterware calls. The environment must be created again before use. You cannot delete an environment and then set new parameters.

Parameters

- env—The environment whose contents are deleted

Returns

environment status enum.

Syntax

```
CLSCRS_ENVCODE clscrs_env_delete(clscrs_env env);
```

clscrs_env_format

Creates a callback with formatted environment lines.

Parameters

- env—Env to dump
- msgf—Callback function
- msgp—Argument to msgp, uninterpreted

Returns

None.

Syntax

```
void clscrs_env_format( clscrs_env env, clscrs_msgf msgf, void *msgp );
```

clscrs_start_resource

Direct Oracle Clusterware to start a named set of resources. If a host is specified, then start there, otherwise start according to the placement policy for the resources in question. If the flag is async, then msgf is never called and Oracle returns the OK status after initiating the call to Oracle Clusterware, and the actual starts are performed asynchronously, and msgf will never be called. If flag is not async and

msgf is not NULL, then msgf will be driven one line at a time with collected output from the start programs.

Parameters

- ctx—The context
- names—The resource(s) to start
- num_names—The number of resources in names[] to start
- host—The host name on which to start the resource, may be NULL
- env—Environment arguments to start
- msgf—User message callback, may be NULL
- msgp—User callback argument, may be NULL
- msgf—Callback function
- msgf—Callback function

Returns

status enum

Syntax

```
CLSCRS_STAT clscrs_start_resource( clscrs_ctx *ctx, const oratext *name[], sword
 num_names, const oratext *host, clscrs_env env, clscrs_msgf msgf, void *msgp,
 uword flag );
```

clscrs_stop_resource

Directs Oracle Clusterware to stop a named set of resources. The flag enables async stop.

Parameters

- ctx— the context
- names—The resource(s) to stop
- num_names—Number of resources in names[] to stop
- flag—Async, force options

Returns

status enum

Syntax

```
clscrs_stop_resource( clscrs_ctx *ctx, const oratext *names[], sword num_names,
clscrs_env env, clscrs_msgf msgf, void *msgarg, uword flag );
```

clscrs_check_resource

Direct Oracle Clusterware to run the check actions for the identified resources. A delay can be specified to indicate a time to wait in seconds before the check action is performed on the server. This occurs after the call returns. This can be used to let something run before performing the check. For each resource identified by the in_splist, a status will be returned in the op_status, indicating whether the resource

exists and whether the caller has permission to invoke operations on the resource. A resource that is explicitly named with no pattern matching will have a NOTFOUND error. A pattern that has no matches will return no error. There will be no valid attributes returned for the resources. There is no way of knowing exactly when the checks will actually be run or be completed. Returns SUCCESS if the operation status of all of the identified resources is SUCCESS, otherwise FAILURE and specific status will be found in the op_status entries. The caller must create the op_status list before the call and destroy it when done.

Parameters

- in_splist [in]—List of resources to check
- delay_secs [in]—Time to wait before the check is run, as appropriate to the resource
- flags [in]—None
- op_status [out]—Resource list structure holding the status of the check operation for each resource

Returns

status enum.

Syntax

```
clscrs_check_resource( clscrs_splist *in_splist, ub4 delay_seconds, uword flags,
 clscrs_reslist *op_status);
```

clscrs_register_resource

Register the resources in the input resource list. The attributes for the resources are encapsulated in the input resource list. The output op_status list contains the results of the register operation for each resource and contains no valid resource attributes.

The caller must create and populate the in_reslist and must create the op_status list. Both the lists must be destroyed by the caller. The op_status list cannot be reused with another API call, you must create and destroy it for each API call.

One or more attributes of an already registered resource can be modified by passing the CLSCRS_FLAG_REG_UPDATE flag. The flags apply to all resources in the input reslist.

Parameters

- in_reslist [in]—List of resources to be registered
- flags [in]—CLSCRS_FLAG_REG_UPDATE or 0
- op_status [out]—Resource list holding the status of the register operation for each resource

Returns

- CLSCRS_STAT_SUCCESS—If all input resources are successfully registered
- CLSCRS_STAT_FAIL—If atleast one resource could not be registered
- CLSCRS_STAT_CONNECTION—If there is a communication error with the crsd process

Syntax

```
clscrs_register_resource(clscrs_reslist *in_reslist, uword flags, clscrs_reslist
*op_status);
```

clscrs_unregister_resource

Unregisters the resources in the input list of resource names. The output `op_status` list contains the results of the unregister operation for each resource. The caller must create and populate the `rqlist` and must create the `op_status` list. Both the lists must be destroyed by the caller. The `op_status` list cannot be reused with another API call. You must create and destroy it for each API call.

Parameters

- `rqlist [in]`—List of resources names to be unregistered
- `flags [in]`—Option flags
- `op_status [out]`—Resource list holding the status of the unregister operation for each resource

Returns

- `CLSCRS_STAT_SUCCESS`—If all input resources are successfully unregistered
- `CLSCRS_STAT_FAIL`—If atleast one resource could not be unregistered
- `CLSCRS_STAT_CONNECTION`—If there is a communication error with the `crsd` process

Syntax

```
clscrs_unregister_resource(clscrs_splist *rqlist, uword flags, clscrs_reslist *op_
status);
```

clscrs_stat

Obtains information about the resources identified in `rqlist`. The resources can be restricted to a specific node given in the nodename parameter. The information is returned in the output `out_reslist`.

If the `rqlist` is `NULL`, information about all registered resources is returned. If the node name is not `NULL`, the result is restricted to resources running on the specific node. Otherwise, all nodes in the cluster are considered.

The `res_flags` are used to scope the resource domain that is matched. There are no values for this parameter. The `res_attrs` identifies the attributes to be returned in the `out_reslist`. The caller may create a specific list of attributes of interest or use a predefined list returned by one of the macros `CLSCRS_ATTRS_NONE`, `CLSCRS_ATTRS_ALL`, `CLSCRS_ATTRS_STATE`, `CLSCRS_ATTRS_SHORT`. The attribute list applies to all resources identified by the `rqlist`. Any errors for a resource are returned in the output `out_reslist`.

Parameters

- `rqlist [in]`—List of resource names to query
- `nodename [in]`—Query resources running only on the given node (optional, may be `NULL`)
- `res_attrs [in]`—List to specify the attributes to be returned for each resource

- res_flags [in]—Flags to restrict the resources being matched
- out_reslist [out]—List holding the returned resource information

Returns

- CLSCRS_STAT_SUCCESS if information is obtained for all input resources
- CLSCRS_STAT_FAIL if no information is available for atleast one resource
- CLSCRS_STAT_CONNECTION if there is an error communicating with the crsd process

Syntax

```
clscrs_stat(clscrs_splist *rqlist, oratext *nodename, uword res_flags, clscrs_
splist *res_attrs, clscrs_reslist *out_reslist);
```

Functions for Managing Resource Structures

This section describes functions for creating, managing, and destroying resources and the attributes associated with resources. The following structures are used in this section:

- clscrs_sp—This is a stringpair (sp) structure. It contains a name and a value string. The value may be NULL. It is created and destroyed and its contents can be examined and the value replaced. An sp may be a member of exactly one stringpair list (splist).
- clscrs_splist—This stringpair list (splist) is a list of zero or more stringpairs used in various contexts. Sometimes the list contains names and values, sometimes the list contains only names. The list is created and destroyed; members can be retrieved by name, added, deleted, replaced, and the list can be traversed with first and next operations.
- clscrs_res—This represents a resource abstraction that contains the name of a resource and additional data about the context in which it is used. Sometimes it contains resource attribute data and other times it contains status and return message about an operation. A resource may be in exactly one resource list.
- clscrs_reslist—This resource list is an abstraction used to contain information about zero or more resources. A resource list is created, appended, iterated over, and destroyed.

Export Operations

This section describes the following export operations:

- Stringpair Operations
- Splist Operations
- Resource Operations
- Resource List Operations

Stringpair Operations

This section describes the stringpair operations.

clscrs_sp_set Changes the value part of a stringpair. The new value may be NULL. After the call returns, the memory of the value can be recycled.

Parameters

- sp [in]—Stringpair for which you want to set a value
- value [in]—Value component of the stringpair (may be NULL)

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the sp argument is NULL

Syntax

```
clscrsret clscrs_sp_set(clscrs_sp *sp, const oratext *value);
```

clscrs_sp_get Gets the name and value components of a stringpair.

Parameters

- sp [in]—Stringpair for which the name and value are being obtained
- name [out]—Name component of the stringpair
- value [out]—Value component of the stringpair (may be NULL)

Returns

- clscrsretSUCC on success
- clscrsretNOMEM if no memory can be allocated
- clscrsretBADARG if the sp argument is NULL

Syntax

```
clscrsret clscrs_sp_get(clscrs_sp *sp, oratext **name, oratext **value);
```

clscrs_sp_get_value Gets the value component of a stringpair.

Parameters

- sp [in]—Stringpair for which the name and value are being obtained
- value [out]—Value component of the stringpair (may be NULL)

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the sp argument is NULL

Syntax

```
clscrsret clscrs_sp_get_value(clscrs_sp *sp, oratext **value);
```

Splist Operations

This section describes the splist operations.

clscrs_splist_create Creates a new stringpair list. The memory for the splist is allocated by the function.

Parameters

- ctx [in]—clscrs context
- splist [out]—The new stringpair list created

Returns

- clscrsretSUCC on success
- clscrsretNOMEM if no memory can be allocated
- clscrsretBADCTX if the context is NULL

Syntax

```
clscrsret clscrs_splist_create(clscrs_ctx *ctx, clscrs_splist **splist);
```

clscrs_splist_create_and_set Creates a new stringpair list (splist) and set the name and value for the first stringpair in the list. The memory for the splist is allocated by the function.

Parameters

- ctx [in]—clscrs context
- name [in]—Name component of the stringpair
- value [in]—Value component of the stringpair (may be NULL)
- sp [out]—The new stringpair created

Returns

- clscrsretSUCC on success
- clscrsretNOMEM if no memory can be allocated
- clscrsretBADCTX if the context is NULL
- clscrsretBADARG if the name argument is NULL

Syntax

```
clscrsret clscrs_splist_create_and_set(clscrs_ctx *ctx, const oratext
*name, const oratext *value, clscrs_splist **splist);
```

clscrs_splist_append Adds a new stringpair (sp) to a stringpair list (splist).

Parameters

- splist [in]—splist to add the new {name, value} stringpair.
- name [in]—Name component of the stringpair
- value [in]—Value component of the stringpair (may be NULL)

Returns

- clscrsretSUCC on success
- clscrsretNOMEM if no memory can be allocated
- clscrsretBADARG if the name argument is NULL

Syntax

```
clscrsret clscrs_splist_append(clscrs_splist *splist, const oratext *name, const
oratext *value);
```

clscrs_splist_first Gets the first stringpair (sp) from a stringpair list (splist).

Parameters

- name [in]—splist to get the first sp from
- sp [out]—The first sp from the given splist

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the splist is NULL
- clscrsretEMPTY if there are no stringpair elements in the splist

Syntax

```
clscrsret clscrs_splist_first(clscrs_splist *splist, clscrs_sp **sp);
```

clscrs_splist_next Gets the current next stringpair (sp) from a stringpair list (splist). This function is called to iterate over the stringpairs in a splist.

Parameters

- name [in]—splist to get the next sp from
- sp [out]—The next sp {name, value} in the given splist

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the splist is NULL
- clscrsretENDLIST if there are no more stringpair elements in the splist

Syntax

```
clscrsret clscrs_splist_next(clscrs_splist *splist, clscrs_sp **sp);
```

clscrs_splist_replace Replaces the value for a stringpair (sp) in a stringpair list (splist).

Parameters

- splist [in]—Splist to add the new {name, value} stringpair.
- name [in]—Name for which the value is being replaced.
- value [in]—Replacement value for the name (may be NULL)

Returns

- clscrsretSUCC on successclscrsretBADARG if the name argument is NULL
- clscrsretBADARG if the name argument is NULL
- clscrsretBADARG if the name argument is NULL

Syntax

```
clscrsret clscrs_splist_replace(clscrs_splist *splist, const oratext *name, const
oratext *value);
```

clscrs_splist_delete_sp Deletes a stringpair (sp) from a stringpair list (splist).

Parameters

- splist [in]—Splist from which to delete the {name, value} stringpair.
- splist [in]—Name to be deleted from the given splist.

Returns

- clscrsretSUCC on success
- clscrsretNONAME if there is no stringpair matching the given name
- clscrsretBADARG if the name argument is NULL

Syntax

```
clscrsret clscrs_splist_delete_sp(clscrs_splist *splist, const oratext *name);
```

clscrs_splist_find Finds the value for a stringpair (`sp`) in a stringpair list (`splist`).

Parameters

- `splist [in]`—`Splist` to look into
- `name [in]`—Name for which the value is looked up
- `value [out]`—Value associated with the given name in the given `splist`

Returns

- `clscrsretSUCC` on success
- `clscrsretNONAME` if there is no stringpair matching the given name
- `clscrsretBADARG` if the name argument is `NULL`

Syntax

```
clscrsret clscrs_splist_find(clscrs_splist *splist, const oratext *name, oratext
**value);
```

clscrs_splist_count Counts the number of stringpairs (`sp`) in a stringpair list (`splist`).

Parameters

- `splist [in]`—`Splist` to count the number of stringpairs.
- `count [out]`—The number of stringpairs in the given `splist`.

Returns

- `clscrsretSUCC` on success
- `clscrsretBADARG` if the splist is `NULL`

Syntax

```
clscrsret clscrs_splist_count(clscrs_splist *splist, ub4 *count);
```

clscrs_splist_destroy Frees the memory for a stringpair list (`splist`).

Parameter

- `splist [in]`—`Splist` to destroy

Returns

- `clscrsretSUCC` on success

Syntax

```
clscrsret clscrs_splist_destroy(clscrs_splist **splist);
```

clscrs_res_create Creates a new resource. The memory for the resource structure is allocated by the function. The memory is freed when a resource list (`clscrs_reslist`) is destroyed through `clscrs_reslist_destroy()`.

Parameters

- `ctx [in]`—`clscrs` context
- `resname [in]`—Name of the resource
- `res [out]`—The new resource created

Returns

- clscrsretSUCC on success
- clscrsretNOMEM if no memory can be allocated
- clscrsretBADCTX if the context is NULL
- clscrsretBADARG if the resource name is NULL

Syntax

```
clscrsret clscrs_res_create(clscrs_ctx *ctx, const oratext *resname, clscrs_res
**res);
```

clscrs_res_get_name Gets the name of a resource.

Parameters

- res [in]—Resource for which the name is obtained
- name [out]—Name of the resource

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the resource argument is NULL

Syntax

```
clscrsret clscrs_res_get_name(clscrs_res *res, oratext **name);
```

clscrs_res_set_attr Sets a resource attribute for a resource.

Paramters

- res [in]—Resource for which the attribute is set
- attrname [in]—Name of the resource attribute
- value [in]—Value for the resource attribute (may be NULL)

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the attribute name is NULL
- clscrsretNOMEM if no memory can be allocated

Syntax

```
clscrsret clscrs_res_set_attr(clscrs_res *res, const oratext *attrname, const
oratext *value);
```

clscrs_res_get_attr Gets a resource attribute for a resource.

Parameters

- res [in]—Resource for which the attribute is obtained
- attrname [in]—Name of the resource attribute
- value [out]—Value for the resource attribute

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the attribute name is NULL

- clscrsretNOMEM if no memory can be allocated

Syntax

```
clscrsret clscrs_res_get_attr(clscrs_res *res, const oratext *attrname, oratext
**value);
```

clscrs_res_get_attr_list Gets the attribute list for a resource. The attribute list is a list of stringpairs. The client does not allocate the memory for attribute list.

Paramters

- res [in]—Resource for which the attribute list
- attrlist [out]—List of attributes for the given resource

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the resource is NULL
- clscrsretNOMEM if no memory can be allocated
- clscrsretNOATTRS if there are no attributes set for the resource

Syntax

```
clscrsret clscrs_res_get_attr_list(clscrs_res *res, clscrs_splist **attrlist);
```

clscrs_res_set_attr_list Sets the attribute list for a resource. The attribute list is a list of stringpairs. The list is created from the clscrs_splist_create call.

Parameters

- res [in]—Resource for which the attribute list is set
- attrlist [in]—List of attributes to be set for the given resource

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the resource is NULL

Syntax

```
clscrsret clscrs_res_set_attr_list(clscrs_res *res, clscrs_splist *attrlist);
```

clscrs_res_attr_count Gets the number of attributes for a resource.

Parameters

- res [in]—Resource for which number of attributes is obtained
- count [out]—Number of attributes for the given resource

Returns

- clscrsretSUCC on success
- clscrsretBADARG if the resource is NULL

Syntax

```
clscrsret clscrs_res_attr_count(clscrs_res *res, ub4 *count);
```

clscrs_res_get_op_status Gets the status of an operation for a resource. The memory for the msg is allocated by the function.

Parameters

- `res [in]`—Resource for which the operation status is obtained
- `status [out]`—Status of an operation on the given resource
- `msg [out]`—Text message for the result of an operation on the resource

Returns

- `clscrsretSUCC` on success
- `clscrsretNOMEM` if no memory can be allocated
- `clscrsretNOMSG` if there is no msg available
- `clscrsretBADARG` if the resource is `NULL`

Syntax

```
clscrsret clscrs_res_get_op_status(clscrs_res *res, CLSCRS_STAT *status, oratext
**msg);
```

clscrs_res_get_registered Gets the registration status of a resource.

Parameters

- `res [in]`—Resource for which the operation status is set
- `registered [out]`—Boolean indicating whether the resource is registered

Returns

- `clscrsretSUCC` on success
- `clscrsretNOMEM` if no memory can be allocated
- `clscrsretBADARG` if the resource is `NULL`

Syntax

```
clscrsret clscrs_res_get_registered(clscrs_res *res, boolean *registered);
```

clscrs_res_get_node_list Gets the nodelist currently hosting the resource, or `NULL` if there is no host for the resource or there are no attributes. The caller need not allocate memory for the nodelist.

Parameters

- `res [in]`—Resource for which the nodelist is obtained
- `nodelist [out]`—`Splist` holding the node(s)

Returns

- `clscrsretSUCC` on success
- `clscrsretBADARG` if the resource is `NULL`

Syntax

```
clscrsret clscrs_res_get_node_list(clscrs_res *res, clscrs_splist **nodelist);
```

clscrs_res_destroy Frees the memory for a resource.

Parameter

- `res [in]`—Resource for which the memory is freed

Returns

- clscrsretSUCC on success

Syntax

```
clscrsret clscrs_res_destroy(clscrs_res **res);
```

clscrs_reslist_create Creates a new resource list. The memory for the resource list is allocated by the function.

Parameters

- ctx [in]—clscrs context
- reslist [out]—Resource list (empty) that is created

Returns

- clscrsretSUCC on success
- clscrsretNOMEM if no memory can be allocated
- clscrsretBADCTX if the context is NULL

Syntax

```
clscrsret clscrs_reslist_create(clscrs_ctx *ctx, clscrs_reslist **reslist);
```

clscrs_reslist_append Adds a resource to a resource list.

Parameters

- reslist [in]—Resource list to add the resource to
- res [in]—Resource to add

Returns

- clscrsretSUCC on success
- clscrsretBADARG if reslist is NULL
- clscrsretRESEXISTS if the resource already exists in reslist

Syntax

```
clscrsret clscrs_reslist_append(clscrs_reslist *reslist, clscrs_res *res);
```

clscrs_reslist_first Gets the first resource on a resource list.

Parameters

- reslist [in]—Resource list for which the first resource is obtained
- res [out]—The first resouce on the resource list

Returns

- clscrsretSUCC on success.
- clscrsretBADARG if reslist is NULL
- clscrsretEMPTY if there are no resources in the list

Syntax

```
clscrsret clscrs_reslist_first(clscrs_reslist *reslist, clscrs_res **res);
```

clscrs_reslist_next Gets the current next resource from the resource list. This function is called to iterate over the resources in a resource list.

Parameters

- reslist [in]—Resource list for which the first resource is obtained
- res [out]—The next resouce on the resource list

Returns

- clscrsretSUCC on success
- clscrsretBADARG if reslist is NULL
- clscrsretENDLIST if there are no more resources in the list

Syntax

```
clscrsret clscrs_reslist_next(clscrs_reslist *reslist, clscrs_res **res);
```

clscrs_reslist_find Finds a resource in a resource list.

Parameters

- reslist [in]—Resource list (empty) that is created
- name [in]—Name of the resource that is being obtained
- res [out]—The resouce corresponding to the given name

Returns

- clscrsretSUCC on success
- clscrsretNORES if the resource is not found
- clscrsretBADARG if reslist or name is NULL

Syntax

```
clscrsret clscrs_reslist_find(clscrs_reslist *reslist, const oratext *name,
clscrs_res **res);
```

clscrs_reslist_count Counts the number of resources in a resource list.

Parameters

- reslist [in]—Resource list for which the count is obtained
- count [out]—Number of resources in the resource list

Returns

- clscrsretSUCC on success
- clscrsretBADARG if reslist is NULL

Syntax

```
clscrsret clscrs_reslist_count(clscrs_reslist *reslist, ub4 *count);
```

clscrs_reslist_delete_res Deletes a resource from a resource list.

Parameters

- reslist [in]—Resource list from which to delete the resource
- name [in]—Name of the resource to delete

Returns

- clscrsretSUCC on success
- clscrsretBADARG if reslist or name is NULL
- clscrsretNORES if the resource is not in reslist

Syntax

```
clscrsret clscrs_reslist_delete_res(clscrs_reslist *reslist,
```

clscrs_reslist_destroy Frees the memory for a resource list.

Parameters

- `reslist [in]`—Resource list for which the memory is to be freed

Returns

- `clscrsretSUCC` on success

Syntax

```
clscrsret clscrs_reslist_destroy(clscrs_reslist **reslist);
```

clscrs_get_error_message Retrieves the error message corresponding to the return codes from a `clscrs` API. The memory for the error message is allocated by the caller. If the buffer is not large enough, the length is returned in `msg_len`.

Parameters

- `ctx [in]`—`clscrs` context
- `err_code [in]`—Error code returned from the `clscrs` API
- `msg [out]`—Message corresponding to `err_code`
- `msg_len [inout]`—Length of the message buffer.

Returns

- `clscrsretSUCC` on success

Syntax

```
clscrsret clscrs_get_error_message(clscrs_ctx *ctx, clscrsret err_code, oratext
*msg, sb4 msg_len);
```

clscrs_get_fixed_attrlist Gets the list of attributes corresponding to an attribute group identifier.

Parameters

- `ctx [in]`—`clscrs` context
- `attrgrp [in]`—Attribute group that identifies a group of attributes
- `attrlist [out]`—List of attributes returned that corresponds to the attribute group

Returns

- `clscrsretSUCC` on success

Syntax

```
clscrs_splist* clscrs_get_fixed_attrlist(clscrs_ctx *ctx, clscrs_attr_grp
attrgrp);
```

Resource Operations

This section describes resource functions. The `clscrs_res` resource abstraction contains the name of a resource and additional data appropriate to the context in which it is used. Sometimes it carries status and error return information about an operation. Other times it contains attribute data as input to an operation. A resource

may be in exactly one resource list. If so, its successor may be found with the NEXT operation.

CLSCRSRET clscrs_res_create(clscrs_cts *ctx, const oratext *resname, clscrs_res **res)

Creates a single resource and fills in a handle to it. The resname must be provided, and cannot be NULL.

CLSCRSRET clscrs_res_get_name (clscrs_res *res, oratext **name);

Returns a pointer to the name of a resource. The returned name pointer is only valid as long as the resource exists.

CLSCRSRET clscrs_res_get_op_status (clscrs_res *res, CLSCRS_STAT *stat, oratext **msg)

If there is a valid operation error value in the resource, fills in the stat and the pointer to an error message and returns SUCCESS. May fill in a NULL for the msg. If there is no valid operation status, returns INVALID.

clscrs_splist *clscrs_res_get_node_list(clscrs_res *res);

Returns an splist holding the nodes currently hosting the resource, or NULL if there is no host for the resource or there are no attributes. The count of the list may obtained and the list iterated. The list is owned by the resource and will be destroyed when the resource is destroyed. This operation is a special case, interpreting the semantics of the attribute or attributes that may hold the current hosting member list. There is no specified ordering of the list.

CLSCRSRET clscrs_res_get_attr(clscrs_res *res, const oratext *attrname, oratext **value);

Fills in a pointer to the value of the attribute with the given name and returns SUCCESS if found, or FAILURE if the name is not present in the attribute set of the resource, and INVALID if there is no attribute list in the resource. May assert if given an invalid resource handle.

CLSCRSRET clscrs_res_set_attr(clscrs_res *res, oratext *attrname, oratext *value);

Sets the value of the attribute with the given name and returns SUCCESS or FAILURE if the name is not present in the attribute set of the resource. If the attribute already exists, then its current value will be replaced. On return, the memory for the name and value may be recycled.

CLSCRSRET clscrs_res_attr_count(clscrs_res *res, ub4 *count);

Fills in the number of attributes that will be returned with a scan using an attrIter. Returns SUCCESS if there are attributes. If there are no attributes, returns INVALID, but still sets the count to zero.

CLSCRSRET clscrs_res_get_attr_list(clscrs_res *res, clscrs_splist **attrlist);

Returns an splist for the attributes of the resource, allowing next() operations for a scan. The list is owned by the resource, which will destroy it when the resource is destroyed. Returns SUCCESS if there is an attribute list, or INVALID if there is not one in the resource. There is no specified ordering of the list.

Resource List Operations

This section describes the resource list operations. The `clscrs_reslist` resource list is an abstraction that contains information about zero or more resources. A list is created, appended, iterated over, and destroyed.

CLSCRSRET clscrs_reslist_create(clscrs_ctx *ctx, clscrs_reslist **reslist)

Creates a resource list and fills a handle.

CLSCRSRET clscrs_reslist_count(clscrs_reslist *reslist, ub4 *count)

Fills a count of the number of resources in a list.

CLSCRSRET clscrs_reslist_first(clscrs_reslist *reslist, clscrs_res **first)

Fills a handle to the first resource in the list of resources. If the list is empty, fill in `NULL`.

CLSCRSRET clscrs_reslist_next(clscrs_reslist *reslist, clscrs_res **next)

Fills a handle to the next resource in the list of resources. If the list is empty, fill in `NULL`.

CLSCRSRET clscrs_reslist_find(clscrs_reslist *reslist, const oratext *name, clscrs_res **res)

Finds a resource by name in a `reslist`, and fill in a handle to the one located. If the resource is not found, fill in `NULL`. Does an exact match lookup with no pattern matching.

CLSCRS_RES *clscrs_reslist_destroy(clscrs_res *res)

Deletes a resource list and all of the resources that it currently contains.

E

Oracle Cluster Registry Configuration Tool Command Reference

This appendix describes the syntax of the Oracle Cluster Registry (OCR) tool, OCRCONFIG.

You use the `ocrconfig` command to perform OCR Configuration Tool operations with administrative privileges on Linux or UNIX systems, or as a user with Administrator privileges on Windows systems. The `ocrconfig` command syntax is as follows where *`options`* is one of the verbs shown in the Option column of Table E–1:

```
ocrconfig -option
```

Table E–1 The ocrconfig Command Options

Option	Purpose
`-backuploc` *pathname*	Specifies an OCR backup directory location. For this entry, specify a full path name that is accessible by all of the nodes. **Note:** The default location for generating backups on Linux or UNIX systems is *`CRS_home`*`/cdata/`*`cluster_name`* where *`cluster_name`* is the name of your cluster. The Windows default location for generating backups uses the same path structure.
`-downgrade`	Downgrades an OCR to an earlier version.
`-export`	Exports the contents of an OCR into a target file.
`-help`	Displays help for the `ocrconfig` commands.
`-import`	Imports the OCR contents from a previously *exported* OCR file.
`-manualbackup`	Performs an OCR backup on demand. The ocrconfig command creates the backup in the location that you specify with the `-backuploc` option.
`-overwrite`	Updates an OCR configuration that is recorded on the OCR with the current OCR configuration information that is found on the node from which you are running this command.
`-repair`	Updates an OCR configuration on the node from which you are running this command with the new configuration information specified by this command.
`-replace`	Adds, replaces, or removes an OCR location.
`-restore`	Restores an OCR from an automatically created OCR backup file.

Table E–1 (Cont.) The ocrconfig Command Options

Option	Purpose
`-showbackup [auto|manual]`	Displays the backup location, timestamp, and the originating node name of the backup files that Oracle created. By default, the `-showbackup` option displays information for both automatic and manual backups. You can optionally specify the `auto` or the `manual` flag to display information about only automatic backups or only manual backups, respectively: ■ `auto`—Displays information about automatic backups that Oracle created in the past 4 hours, 8 hours, 12 hours, and in the last day and week. ■ `manual`—Displays information about manual backups that you invoke using the `-manualbackup` option. **Note:** You do not have to be the `root` user to issue the `-showbackup` option.
`-upgrade`	Upgrades an OCR to a later version.

For example, to export the OCR contents to a binary file, use the `ocrconfig` command with the following syntax where *`file_name`* is the file to which you want to export the OCR contents as follows:

```
ocrconfig -export file_name
```

OCR Log File Locations

The OCRCONFIG tool creates a log file in *`CRS_home`*`/log/hostname/client`. To change the amount of logging, edit the path in the *`CRS_home`*`/srvm/admin/ocrlog.ini` file.

F

Troubleshooting Oracle Clusterware

This appendix introduces monitoring the Oracle Clusterware environment and explains how you can enable dynamic debugging to troubleshoot Oracle Clusterware processing, and enable debugging and tracing for specific components and specific Oracle Clusterware resources to focus your troubleshooting efforts.

This appendix contains the following topics:

Monitoring Oracle Clusterware

You can use Oracle Enterprise Manager to monitor the Oracle Clusterware environment. When you log in to Oracle Enterprise Manager using a client browser, the Cluster Database Home page appears where you can monitor the status of both Oracle Clusterware environments. Monitoring can include such things as:

- Notification if there are any VIP relocations
- Status of the Oracle Clusterware on each node of the cluster using information obtained through the Cluster Verification Utility (cluvfy)
- Notification if node applications (nodeapps) start or stop
- Notification of issues in the Oracle Clusterware alert log for the OCR, voting disk issues (if any), and node evictions

The Cluster Database Home page is similar to a single-instance Database Home page. However, on the Cluster Database Home page, Oracle Enterprise Manager displays the system state and availability. This includes a summary about alert messages and job activity, as well as links to all the database and Automatic Storage Management (ASM) instances. For example, you can track problems with services on the cluster including when a service is not running on all of the preferred instances or when a service response time threshold is not being met.

You can use the Oracle Enterprise Manager Interconnects page to monitor the Oracle Clusterware environment. The Interconnects page shows the public and private interfaces on the cluster, the overall throughput on the private interconnect, individual throughput on each of the network interfaces, error rates (if any) and the load contributed by database instances on the interconnect, including:

- Overall throughput across the private interconnect
- Notification if a database instance is using public interface due to misconfiguration
- Throughput and errors (if any) on the interconnect
- Throughput contributed by individual instances on the interconnect

All of this information also is available as collections that have a historic view. This is useful in conjunction with cluster cache coherency, such as when diagnosing problems related to cluster wait events. You can access the Interconnects page by clicking the Interconnect tab on the Cluster Database home page.

Also, the Oracle Enterprise Manager Cluster Database Performance page provides a quick glimpse of the performance statistics for a database. Statistics are rolled up across all the instances in the cluster database in charts. Using the links next to the charts, you can get more specific information and perform any of the following tasks:

- Identify the causes of performance issues.
- Decide whether resources need to be added or redistributed.
- Tune your SQL plan and schema for better optimization.
- Resolve performance issues

The charts on the Cluster Database Performance page include the following:

- **Chart for Cluster Host Load Average**—The Cluster Host Load Average chart in the Cluster Database Performance page shows potential problems that are outside the database. The chart shows maximum, average, and minimum load values for available nodes in the cluster for the previous hour.
- **Chart for Global Cache Block Access Latency**—Each cluster database instance has its own buffer cache in its System Global Area (SGA). Using Cache Fusion, Oracle RAC environments logically combine each instance's buffer cache to enable the database instances to process data as if the data resided on a logically combined, single cache.
- **Chart for Average Active Sessions**—The Average Active Sessions chart in the Cluster Database Performance page shows potential problems inside the database. Categories, called wait classes, show how much of the database is using a resource, such as CPU or disk I/O. Comparing CPU time to wait time helps to determine how much of the response time is consumed with useful work rather than waiting for resources that are potentially held by other processes.
- **Chart for Database Throughput**—The Database Throughput charts summarize any resource contention that appears in the Average Active Sessions chart, and

also show how much work the database is performing on behalf of the users or applications. The Per Second view shows the number of transactions compared to the number of logons, and the amount of physical reads compared to the redo size for each second. The Per Transaction view shows the amount of physical reads compared to the redo size for each transaction. Logons is the number of users that are logged on to the database.

In addition, the Top Activity drilldown menu on the Cluster Database Performance page enables you to see the activity by wait events, services, and instances. Plus, you can see the details about SQL/sessions by going to a prior point in time by moving the slider on the chart.

See Also: *Oracle Database 2 Day + Real Application Clusters Guide*

Dynamic Debugging

You can use `crsctl` commands as the `root` user to enable dynamic debugging for Oracle Clusterware, the Event Manager (EVM), and the clusterware subcomponents. You can dynamically change debugging levels using `crsctl` commands. Debugging information remains in the Oracle Cluster Registry (OCR) for use during the next startup. You can also enable debugging for resources.

The `crsctl` syntax to enable debugging for Oracle Clusterware is:

```
crsctl debug log crs "CRSRTI:1,CRSCOMM:2"
```

The `crsctl` syntax to enable debugging for EVM is:

```
crsctl debug log evm "EVMCOMM:1"
```

The `crsctl` syntax to enable debugging for resources is:

```
crsctl debug log res "resname:1"
```

Component Level Debugging

You can use `crsctl` commands as the `root` user to enable dynamic debugging for the Oracle Clusterware Cluster Ready Services (CRS), Oracle Cluster Registry (OCR), Cluster Synchronization Services (CSS), and the Event Manager (EVM).

This section contains the following topics:

- Enabling Debugging for CRS, OCR, CSS, and EVM Modules
- Creating an Initialization File to Contain the Debugging Level

Enabling Debugging for CRS, OCR, CSS, and EVM Modules

You can enable debugging for the CRS, OCR, CSS, and EVM modules and their components by setting environment variables or by issuing `crsctl debug` commands using the following syntax:

```
crsctl debug log module_name component:debugging_level
```

You must issue the `crsctl debug` command as the `root` user, and supply the following information:

- *module_name*—The name of the module: CRS, EVM, or CSS.

- *component*—The name of a component for the CRS, OCR, EVM, or CSS module. See Table F–1 for a list of all of the components.
- *debugging_level*—A number from 1 to 5 to indicate the level of detail you want the debug command to return, where 1 is the least amount of debugging output and 5 provides the most detailed debugging output.

  You can dynamically change the debugging level in the crsctl command, or you can configure an init file for changing the debugging level as described in "Creating an Initialization File to Contain the Debugging Level" on page F-5.

The following commands show examples of how to enable debugging for the various modules:

- To enable debugging for Oracle Clusterware:

```
crsctl debug log crs "CRSRTI:1,CRSCOMM:2"
```

- To enable debugging for OCR:

```
crsctl debug log crs "CRSRTI:1,CRSCOMM:2,OCRSRV:4"
```

- To enable debugging for EVM:

```
crsctl debug log evm "EVMCOMM:1"
```

- To enable debugging for resources

```
crsctl debug log res "resname:1"
```

To list the components that can be used for debugging, issue the `crsctl lsmodules` command using the following syntax and supply `crs`, `evm`, or `css` for the *module_name* parameter:

```
crsctl lsmodules module_name
```

Note: You do not have to be the `root` user to run the `crsctl` command with the `lsmodules` option.

Table F–1 shows the components for the CRS, OCR, EVM, and CSS modules, respectively. Note that some of the component names are common between the CRS, EVM, and CSS daemons and may be enabled on that specific daemon. For example `COMMNS` is the NS layer and because each daemon uses the NS layer, you can enable this specific module component on any of the daemons to get specific debugging information.

Table F–1 Components for the CRS, OCR, EVM, and CSS Modules

CRS Modules[1]	OCR Modules[2]	EVM Modules[3]	CSS Modules[4]
CRSUI	OCRAPI	EVMD	CSSD
CRSCOMM	OCRCLI	EVMDMAIN	COMMCRS
CRSRTI	OCRSRV	EVMCOMM	COMMNS
CRSMAIN	OCRMAS	EVMEVT	
CRSPLACE	OCRMSG	EVMAPP	
CRSAPP	OCRCAC	EVMAGENT	
CRSRES	OCRRAW	CRSOCR	
CRSCOMM	OCRUTL	CLUCLS	
CRSOCR	OCROSD	CSSCLNT	
CRSTIMER		COMMCRS	
CRSEVT		COMMNS	
CRSD	OCR Tools Modules		
CLUCLS			
CSSCLNT	OCRCONF		
COMMCRS	OCRDUMP		
COMMNS	OCRCHECK		

[1] List the CRS component modules using the `crsctl lsmodules crs` command.
[2] You cannot list the OCR modules using the `crsctl lsmodules` command.
[3] List the EVM component modules using the `crsctl lsmodules evm` command.
[4] List the CSS component modules using the `crsctl lsmodules css` command.

Creating an Initialization File to Contain the Debugging Level

This section describes how to specify the debugging level in an initialization file. This debugging information is stored for use during the next startup.

For each process that you want to debug, you can create an initialization file that contains the debugging level.

The initialization file name includes the name of the process that you are debugging (*process_name*`.ini`). The file is located in the `|`*Oracle_home*`/log/`*hostname*`/admin/|` directory.

For example, `ORACLE_HOME/log/hostA/admin/clscfg.ini` is the name for the CLSCFG debugging initialization file on hostA.

> **See Also:** "Enabling Debugging for CRS, OCR, CSS, and EVM Modules" on page F-3 for information about dynamically changing debugging levels by specifying the level number (from 1 to 5) on the `crsctl` command

Oracle Clusterware Shutdown and Startup

You can start or stop Oracle Clusterware by issuing `crsctl start` and `stop` commands.

Example 1 Stopping Oracle Clusterware

To stop Oracle Clusterware and its related resources on a specific node, issue the following command:

```
crsctl stop crs
```

Example 2 Starting Oracle Clusterware

To start Oracle Clusterware and its related resources on a specific node, issue the following command:

```
crsctl start crs
```

Note: You must run these `crsctl` commands as the `root` user.

Enabling and Disabling Oracle Clusterware Daemons

When the Oracle Clusterware daemons are enabled, they start automatically at the time the node is started. To prevent the daemons from starting, you can disable them using `crsctl` commands. You can use `crsctl` commands as follows to enable and disable the startup of the Oracle Clusterware daemons.

Issue the following command to enable startup for all of the Oracle Clusterware daemons:

```
crsctl enable crs
```

Issue the following command to disable the startup of all of the Oracle Clusterware daemons:

```
crsctl disable crs
```

Note: You must run these `crsctl` commands as the `root` user.

Determining the Active Versions and Software Versions

You can determine the active version or the software version running on the local node cluster by issuing `crsctl activeversion` and `softwarewareversion` commands.

- The software version is the binary version of the software on a particular cluster node.
- The active version is the lowest software version running in a cluster.

These versions are used while upgrading a cluster.

Example 1 Determining the Active Version

To determine the active version on the local node, issue the following command:

```
crsctl query crs activeversion
```

Example 2 Determining the Software Version

To determine the software version on the local node, issue the following command:

```
crsctl query crs softwareversion
```

Diagnostics Collection Script

Every time an Oracle Clusterware error occurs, you should use run the `diagcollection.pl` script to collect diagnostic information from Oracle Clusterware in trace files. The diagnostics provide additional information so Oracle Support can resolve problems. Run this script from the following location:

CRS_home`/bin/diagcollection.pl`

Note: You must run this script as the `root` user.

Oracle Clusterware Alerts

Oracle Clusterware posts alert messages when important events occur. The following is an example of an alert from the CRSD process:

```
[NORMAL] CLSD-1201: CRSD started on host %s
[ERROR] CLSD-1202: CRSD aborted on host %s. Error [%s]. Details in %s.
[ERROR] CLSD-1203: Failover failed for the CRS resource %s. Details in %s.
[NORMAL] CLSD-1204: Recovering CRS resources for host %s
[ERROR] CLSD-1205: Auto-start failed for the CRS resource %s. Details in %s.
```

The location of this alert log on Linux, UNIX, and Windows systems is in the following directory path, where *CRS_home* is the name of the location of Oracle Clusterware: *CRS_home*`/log/hostname/alerthostname.log`.

The following example shows an EVMD alert:

```
[NORMAL] CLSD-1401: EVMD started on node %s
[ERROR] CLSD-1402: EVMD aborted on node %s. Error [%s]. Details in %s.
```

Resource Debugging

You can use `crsctl` command to enable resource debugging using the following syntax:

```
crsctl debug log res "ora.node1.vip:1"
```

This has the effect of setting the environment variable `USER_ORA_DEBUG`, to `1`, before running the `start`, `stop`, or `check` action scripts for the `ora.node1.vip` resource.

Note: You must run this `crsctl` command as the `root` user.

Checking the Health of the Clusterware

Use the `crsctl check` command to determine the health of your clusterware as in the following example:

```
crsctl check crs
```

Issue the following command to determine the health of individual daemons where *daemon* is `crsd`, `cssd` or `evmd`:

```
crsctl check daemon
```

Note: You do not have to be the `root` user to perform health checks.

Clusterware Log Files and the Unified Log Directory Structure

Oracle uses a unified log directory structure to consolidate the Oracle Clusterware component log files. This consolidated structure simplifies diagnostic information collection and assists during data retrieval and problem analysis.

Oracle retains five files that are 20MB in size for the CSSD process and one file that is 10MB in size for the `CRSD` and `EVMD` processes. In addition, Oracle deletes the oldest

log file for any log file group when the maximum storage limit for the group's files exceeds 10MB. Alert files are stored in the directory structures shown in Table F–2.

Table F–2 Locations of Oracle Clusterware Component Log Files

Component	Log File Location[1]
Cluster Ready Services Daemon (crsd) Log Files	*CRS_home*/log/*hostname*/crsd
Oracle Cluster Registry (OCR) records l	For the OCR tools (OCRDUMP, OCRCHECK, OCRCONFIG) record log information in the following location:[2] *CRS_Home*/log/*hostname*/client The OCR server records log information in the following location:[3] *CRS_home*/log/*hostname*/crsd
Oracle Processor Daemon (OPROCD)	The following path is specific to Linux[4]: /etc/oracle/*hostname*.oprocd.log
Cluster Synchronization Services (CSS)	*CRS_home*/log/*hostname*/cssd
Event Manager (EVM) information generated by evmd	*CRS_home*/log/*hostname*/evmd
Oracle RAC RACG	The Oracle RAC high availability trace files are located in the following two locations: *CRS_home*/*log*/hostname/racg and $ORACLE_HOME/log/hostname/racg Core files are in subdirectories of the log directory. Each RACG executable has a subdirectory assigned exclusively for that executable. The name of the RACG executable subdirectory is the same as the name of the executable.

1 The directory structure is the same for Linux, UNIX, and Windows systems.

2 To change the amount of logging, edit the path in the *CRS_home*/srvm/admin/ocrlog.ini file.

3 To change the amount of logging, edit the path in the *CRS_home*/log/*hostname*/crsd/crsd.ini file.

4 This path is dependent upon the installed Linux or UNIX platform.

Troubleshooting the Oracle Cluster Registry

This following topics in this section explain how to troubleshoot the OCR:

- Using the OCRDUMP Utility to View Oracle Cluster Registry Content
- Using the OCRCHECK Utility
- Oracle Cluster Registry Troubleshooting

Using the OCRDUMP Utility to View Oracle Cluster Registry Content

This section explains how to use the OCRDUMP utility to view OCR content for troubleshooting. The OCRDUMP utility enables you to view the OCR contents by writing OCR content to a file or stdout in a readable format.

You can use a number of options for OCRDUMP. For example, you can limit the output to a key and its descendents. You can also write the contents to an XML file that you can view using a browser. OCRDUMP writes the OCR keys as ASCII strings and values in a datatype format. OCRDUMP retrieves header information based on a best effort basis.

OCRDUMP also creates a log file in *CRS_home*/log/hostname/client. To change the amount of logging, edit the file *CRS_Home*/srvm/admin/ocrlog.ini.

To change the logging component, edit the entry containing the `comploglvl=` entry. For example, to change the logging of the ORCAPI component to 3 and to change the logging of the OCRRAW component to 5, make the following entry in the `ocrlog.ini` file:

```
comploglvl="OCRAPI:3;OCRRAW:5"
```

> **Note:** Make sure that you have file creation privileges in the *CRS_home* directory before using the OCRDUMP utility.

OCRDUMP Utility Syntax and Options

This section describes the OCRDUMP utility command syntax and usage. Run the `ocrdump` command with the following syntax where *filename* is the name of a target file to which you want Oracle to write the OCR output and where *keyname* is the name of a key from which you want Oracle to write OCR subtree content:

```
ocrdump [file_name|-stdout] [-backupfile backup_file_name] [-keyname keyname]
[-xml] [-noheader]
```

Table F–3 describes the OCRDUMP utility options and option descriptions.

Table F–3 OCRDUMP Options and Option Descriptions

Options	Description
file_name	The name of a file to which you want OCRDUMP to write output. By default, output from the OCRDUMP utility is written to the predefined output file named `OCRDUMPFILE`. The *file_name* option redirects OCRDUMP output to the file that you specify.
`-stdout`	Use this option to redirect the OCRDUMP output to the text terminal that initiated the program. If you do not redirect the output, output from the OCRDUMP utility is written to the predefined output file named `OCRDUMPFILE` by default.
`-keyname`	The name of an OCR key whose subtree is to be dumped.
`-xml`	Writes the output in XML format.
`-noheader`	Does not print the time at which you ran the command and when the OCR configuration occurred.
`-backupfile`	Option to identify a backup file.
backup_file_name	The name of the backup file with the content you want to view. You can query the backups using the `ocrconfig -showbackup` command.

OCRDUMP Utility Examples

The following `ocrdump` utility examples extract various types of OCR information and write it to various targets:

```
ocrdump
```

Writes the OCR content to a file called `OCRDUMPFILE` in the current directory.

```
ocrdump MYFILE
```

Writes the OCR content to a file called `MYFILE` in the current directory.

```
ocrdump -stdout -keyname SYSTEM
```

Writes the OCR content from the subtree of the key `SYSTEM` to `stdout`.

```
ocrdump -stdout -xml
```

Writes the OCR content to `stdout` in XML format.

Sample OCRDUMP Utility Output

The following OCRDUMP examples show the `KEYNAME`, `VALUE TYPE`, `VALUE`, permission set (`user`, `group`, `world`) and access rights for two sample runs of the `ocrdump` command. The following shows the output for the `SYSTEM.language` key that has a text value of `AMERICAN_AMERICA.WE8ASCII37`.

```
[SYSTEM.language]
ORATEXT : AMERICAN_AMERICA.WE8ASCII37
SECURITY : {USER_PERMISSION : PROCR_ALL_ACCESS, GROUP_PERMISSION : PROCR_READ,
 OTHER_PERMISSION : PROCR_READ, USER_NAME : user, GROUP_NAME : group
}
```

The following shows the output for the `SYSTEM.version` key that has integer value of `3`:

```
[SYSTEM.version]
UB4 (10) : 3
SECURITY : {USER_PERMISSION : PROCR_ALL_ACCESS, GROUP_PERMISSION : PROCR_READ,
 OTHER_PERMISSION : PROCR_READ, USER_NAME : user, GROUP_NAME : group
}
```

Using the OCRCHECK Utility

The OCRCHECK utility displays the version of the OCR's block format, total space available and used space, OCRID, and the OCR locations that you have configured. OCRCHECK performs a block-by-block checksum operation for all of the blocks in all of the OCRs that you have configured. It also returns an individual status for each file as well as a result for the overall OCR integrity check.

The following example shows a sample of the OCRCHECK utility output:

```
Status of Oracle Cluster Registry is as follows :
        Version                  :          2
        Total space (kbytes)     :     262144
        Used space (kbytes)      :      16256
        Available space (kbytes) :     245888
        ID                       : 1918913332
        Device/File Name         : /dev/raw/raw1
                                   Device/File integrity check succeeded
        Device/File Name         : /dev/raw/raw2
                                   Device/File integrity check succeeded

        Cluster registry integrity check succeeded
```

OCRCHECK creates a log file in the directory *CRS_home*/log/hostname/client. To change amount of logging, edit the file *CRS_home*/srvm/admin/ocrlog.ini.

Oracle Cluster Registry Troubleshooting

Table F–4 describes common OCR problems with corresponding resolution suggestions.

Table F–4 Common OCR Problems and Solutions

Problem	Solution
Not currently using OCR mirroring and would like to enable it.	Run the `ocrconfig` command with the `-replace` option as described on page 2-4.
An OCR failed and you need to replace it. Error messages in Enterprise Manager or OCR log file.	Run the `ocrconfig` command with the `-replace` option as described on page 2-5.
An OCR has a misconfiguration.	Run the `ocrconfig` command with the `-repair` option as described on page 2-5.
You are experiencing a severe performance effect from OCR processing or you want to remove an OCR for other reasons.	Run the `ocrconfig` command with the `-replace` option as described on page 2-4.
An OCR has failed and before you can fix it, the node need to be rebooted with only one OCR.	Run the `ocrconfig -repair` command to remove the bad ocr file. Oracle Clusterware will not start if it cannot find all OCRs defined.

Enabling Additional Tracing for Oracle Clusterware High Availability

Oracle Support may ask you to enable tracing to capture additional information. Because the procedures described in this section may affect performance, only perform these activities with the assistance of Oracle Support. This section includes the following topics:

- Generating Additional Trace Information for a Running Resource
- Verifying Event Manager Daemon Communications

Generating Additional Trace Information for a Running Resource

To generate additional trace information for a running resource, Oracle recommends that you use `CRSCTL` commands. For example, issue the following command to turn on debugging for resources:

```
$ crsctl debug log res "resource_name:level"
```

For example, to set the value of the `USR_ORA_DEBUG` initialization parameter to 1 for the VIP resource, issue the following command:

```
$ crsctl debug log res ora.cwclu011.vip:1
```

Verifying Event Manager Daemon Communications

The event manager daemons (`evmd`) running on separate nodes communicate through specific ports. To determine whether the `evmd` for a node can send and receive messages, perform the test described in this section while running session 1 in the background.On node 1, session 1 enter:

```
$ evmwatch -A -t "@timestamp @@"
```

On node 2, session 2 enter:

```
$ evmpost -u "hello" [-h nodename]
```

Session 1 should show output similar to the following:

```
$ 21-Jul-2007 08:04:26 hello
```

Ensure that each node can both send and receive messages by executing this test in several permutations.

G

Oracle Clusterware API Messages

This appendix describes the Oracle Clusterware API messages. The Oracle Clusterware commands and APIs are the primary sources of these error messages.

> **See Also:** *Oracle Database Platform Guide for Microsoft Windows* for Windows messages and for all other messages refer search online at
>
> http://tahiti.oracle.com

CRS-0184: Cannot communicate with the CRS daemon.

Cause: The CRS daemon on the local node is either not running or there was an internal communication error with the CRS daemon.

Action: Check if the CRS daemon process is running on the local node.

CRS-0210: Could not find resource '%s'.

Cause: An attempt was made to operate on a resource that is not registered.

Action: Check if the resource is registered using crs_stat.

CRS-0211: Resource '%s' has already been registered.

Cause: An attempt was made to register a resource that is already registered.

Action: Check if the resource is registered using crs_stat.

CRS-0213: Could not register resource '%s'.

Cause: There was an internal error while registering the resource.

Action: Check the CRS daemon log file.

CRS-0214: Could not unregister resource '%s'.

Cause: There was an internal error while unregistering the resource.

Action: Check the CRS daemon log file.

CRS-0215: Could not start resource '%s'.

Cause: There was an internal error while starting the resource.

Action: Check the CRS daemon log file.

CRS-0216: Could not stop resource '%s'.

Cause: There was an internal error while stopping the resource.

Action: Check the CRS daemon log file.

CRS-0217: Could not relocate resource '%s'.

Cause: There was an internal error while relocating the resource.

Action: Check the CRS daemon log file.

CRS-0218: Could not restart the resource '%s' on the original node.

Cause: There was an internal error while restarting the resource.

Action: Check the CRS daemon log file.

CRS-0219: Could not update resource '%s'.

Cause: There was an internal error while updating the resource.

Action: Check the CRS daemon log file.

CRS-0220: Resource '%s' has invalid resource profile.

Cause: Invalid attributes in the resource profile.

Action: Run crs_profile -validate to identify the invalid attributes.

CRS-0221: Resource '%s"s action script cannot be found.

Cause: The action script has been deleted from the file system.

Action: Run crs_stat -p to determine the action script location and to check for its existence.

CRS-0223: Resource '%s' has placement error.

Cause: There was no host available on which to failover/start the resource based on the Placement Policy for the resource.

Action: Check the target host for the resource and restart the resource using the crs_start command.

CRS-0230: Member '%s' is not in the cluster.

Cause: The hostname was not found in the cluster.

Action: Check the hostnames in the cluster.

CRS-0232: Cluster member is down. Cannot perform operation.

Cause: The node on which CRS is attempting to start the resource is down.

Action: Start the node and retry the operation.

CRS-0233: Resource or relatives are currently involved with another operation.

Cause: Another CRS daemon was operating on the same resource.

Action: Wait for a minute and try the command or operation again.

CRS-0253: CRS configuration error, the CRS default directory is not set in OCR.

Cause: The OCR key which contains the user default CRS key is not initialized.

Action: Check the CRS configuration. If necessary reinstall CRS.

CRS-0254: Authorization failure.

Cause: The user permissions were insufficient to operate on the resource.

Action: Check the permissions associated with the resource using crs_getperm.

CRS-0255: CRSD is not running in privileged mode. Insufficient permissions to run this command.

Cause: The CRS daemon was not running as the privileged user.

Action: Check if the CRS daemon is running as root (UNIX or Linux) or Administrator (Windows).

CRS-0256: Username conflicts with the owner of the resource.

Cause: An attempt was made to give separate user level permissions for the owner of the resource.

Action: Check the owner of the resource and the user being given permissions.

CRS-0257: Groupname conflicts with the primary group of the resource.

Cause: An attempt was made to give separate group level permissions for the primary group of the resource.

Action: Check the primary group of the resource and the group being given permissions.

CRS-0258: Invalid ACL string format.

Cause: CRS-258: Invalid ACL string format.

Action: Check the syntax of the permission string (ACL).

CRS-0259: Owner of the resource does not belong to the group.

Cause: The owner of the resource does not belong to the expected group.

Action: If this resource is owned by the root user, check if the root user belongs to the DBA group.

CRS-0402: Could not make safe dir('%s').

Cause: Unable to create safe directory('%s').

Action: Please check if you have proper permissions and sufficient space on the disk to create the directory.

CRS-0403: Could not chdir to safe dir('%s').

Cause: Unable to change directory to safe dir('%s').

Action: Please check if safe dir exists and if you have proper permissions.

CRS-0406: Could not create lock dir ('%s').

Cause: Unable to create lock directory ('%s')

Action: Please check if you have proper permissions and sufficient space on the disk to create the directory.

CRS-0407: Another CRSD may be running, could not obtain lock file '%s'.

Cause: Unable to obtain lock file as another CRSD may be running

Action: Please stop the existing CRSD before attempting to start CRSD again.

CRS-0413: Could not initialize the CSS context.

Cause: Unable to communicate with the cluster services.

Action: Verify that the CSS Daemon is properly configured and is running.

CRS-0414: Could not establish EVM connection.

Cause: Unable to communicate with EVM daemon.

Action: Run the 'crsctl check evmd' command to determine whether EVM daemon is properly configured and is running.

CRS-0451: CRS configuration error, unable to initialize OCR.

Cause: The OCR that contains information about the CRS configuration is unavailable.

Action: Check the CRS configuration. If necessary reinstall CRS.

CRS-0452: CRS configuration error, unable to find CRSD Connection Information in OCR.

Cause: The OCR key which contains the user default CRSD connection is not initialized.

Action: Check the CRS configuration. If necessary reinstall CRS.

CRS-0453: CRS configuration error, unable to find Instance Information in OCR.

Cause: The OCR key which contains the Instance's information is not initialized.

Action: Add the instance using srvctl.

CRS-0471: Node number is not found.

Cause: Cluster Services is unable to retrieve the node name.

Action: Verify your cluster installation, including any vendor cluster ware. If necessary reinstall the cluster.

CRS-0472: Node name is not found.

Cause: Cluster Services is unable to retrieve the node name.

Action: Verify your cluster installation, including any vendor cluster ware. If necessary reinstall the cluster.

CRS-1005: Failed to get required resources.

Cause: There was an internal error while evaluating the required resources for the subject resource.

Action: Check if the status of any resource is `UNKONOWN` using `crs_stat -t`.

CRS-1006: No more members to consider.

Cause: There was no host found on which to start the resource based on the placement policy.

Action: Check the placement policy and the required resources for the subject resource.

CRS-1007: Failed after successful dependency consideration.

Cause: There was no host found on which to start the resource based on the placement policy.

Action: Check the placement policy and the required resources for the subject resource.

CRS-1009: Resource '%s' is already running on member '%s'.

Cause: An attempt was made to start a resource on a host while it is already running on that host.

Action: This is an insignificant error. Check the operation being performed. 1011, 0, Trying to relocate to a dead member.

CRS-2001: User does not have permission to start CRSD.

Cause: Unable to start CRSD due to insufficient permissions.

Action: Start CRSD as a privileged user.

CRS-2007: Could not communicate with EVM.

Cause: Unable to communicate with EVM daemon.

Action: Run the 'crsctl check evmd' command to determine whether EVM daemon is properly configured and is running.

H

Oracle Clusterware Alert Messages

This appendix describes the Oracle Clusterware messages that are returned in the Oracle Clusterware alert log.

> **See Also:** *Oracle Database Platform Guide for Microsoft Windows* for Windows messages and for all other messages refer search online at
>
> http://tahiti.oracle.com

CRS--0600: [%s] Error [%s]. Details in %s.

Cause: None

Action: None

CRS-1001 The OCR was formatted using version %d.

Cause: Successfully formatted the OCR location(s).

Action: None

CRS-1002 The OCR was restored from %s.

Cause: The OCR was successfully restored from a backup file as requested by the user.

Action: None

CRS-1003 The OCR format was downgraded to version %d.

Cause: The OCR was successfully downgraded to an earlier block format as requested by the user.

Action: None

CRS-1004 The OCR was imported from %s.

Cause: Successfully imported the OCR contents from a file as requested by the user.

Action: None

CRS-1005 The OCR upgrade was completed. Version has changed from %d to %d. Details in %s.

Cause: The OCR was successfully upgraded to a newer block format.

Action: None

CRS-1006 The OCR location %s is inaccessible. Details in %s.

Cause: An error occurred while accessing the OCR.

Action: Use the ocrcheck command to validate the accessibility of the device and its block integrity. Check that the OCR location in question has the correct

permissions. Determine whether this behavior is limited to one node or whether it occurs across all of the nodes in the cluster. Use the ocrconfig command with the -replace option to replace the OCR location.

CRS-1007 The OCR/OCR mirror location was replaced by %s.

Cause: The OCR location was successfully replaced as requested by the user.

Action: None

CRS-1008 Node %s is not responding to OCR requests. Details in %s.

Cause: Error in communicating to the OCR server on a peer node. This OCR did not receive a notification regarding its peer's death within the specified time.

Action: Contact Oracle Support.

CRS-1009 The OCR configuration is invalid. Details in %s.

Cause: The OCR configuration on this node does not match the OCR configuration on the other nodes in the cluster.

Action: Determine the OCR configuration on the other nodes in the cluster on which Oracle Clusterware is running by using the ocrcheck command. Run the ocrconfig command with the -repair option to correct the OCR configuration on this node.

CRS-1010 The OCR mirror location %s was removed.

Cause: The OCR location was successfully removed as requested by the user.

Action: None

CRS-1011 OCR cannot determine that the OCR content contains the latest updates. Details in %s.

Cause: The OCR could not be started. The OCR location configured on this node does not have the necessary votes and might not have the latest updates.

Action: Ensure that the other nodes in the cluster have the same OCR location configured. If the configuration on the other nodes in the cluster does not match, then run the ocrconfig command with the -repair option to correct the configuration on this node.

If the configurations on all of the nodes match, use the ocrdump command to ensure that the existing OCR location has the latest updates. Run the ocrconfig command with the -overwrite option to correct the problem. If the se procedures do not correct the problem, then contact Oracle Support Services.

CRS-1012 The OCR service started on node %s.

Cause: The OCR was successfully started.

Action: None

CRS-1013 The OCR at %s was successfully formatted using version %d. Ignore earlier CRS-1006 messages if any.

Cause: Successfully formatted the OCR location(s).

Action: Ignore earlier CRS-1006 messages if any.

CRS-1201 CRSD started on node %s.

Cause: CRSD has started, possibly due to a CRS start, or a node reboot or a CRSD restart.

Action: None Required. You can run the command 'crsctl check crsd' to validate the health of the CRSD

CRS-1202 CRSD aborted on node %s. Error [%s]. Details in %s.

Cause: Fatal Internal Error. Check the CRSD log file to determine the cause.

Action: Determine whether the CRSD gets auto-started.

CRS-1203 Failover failed for the CRS resource %s. Details in %s.

Cause: Failover failed due to an internal error. Examine the contents of the CRSD log file to determine the cause.

Action: None

CRS-1204 Recovering CRS resources for node %s.

Cause: CRS resources are being recovered, possibly because the cluster node is starting up online.

Action: Check the status of the resources using the crs_stat command.

CRS-1205 Auto-start failed for the CRS resource %s. Details in %s.

Cause: This message comes up when the auto-start for the resource has failed during a reboot of the cluster node.

Action: Start the resources using the crs_start command.

CRS-1206 Resource %s went into an UNKNOWN state. Force stop the resource using the crs_stop -f command and restart %s.

Cause: Resource went into an UNKNOWN state because the check or the stop action on the resource failed.

Action: Force stop the resource using the crs_stop -f command and restart the resource

CRS-1207 There are no more restart attempts left for resource %s. Restart the resource manually using the crs_start command.

Cause: The Oracle Clusterware is no longer attempting to restart the resource because the resource has failed and the Oracle Clusterware has exhausted the maximum number of restart attempts.

Action: Use the crs_start command to restart the resource manually.

CRS-1401 EVMD started on node %s.

Cause: EVMD has started either because of a CRS start, a node reboot, or an EVMD restart.

Action: None required. You can run the 'crsctl check evmd' command to validate the health of EVMD.

CRS-1402 EVMD aborted on node %s. Error [%s]. Details in %s.

Cause: EVMD has aborted due to an internal error. Check the EVMD log file to determine the cause.

Action: Determine whether the EVMD is auto-started.

CRS-1601 CSSD Reconfiguration complete. Active nodes are %s.

Cause: A node joined or left the cluster

Action: None

CRS-1602 CSSD aborted on node %s. Error [%s]. Details in %s.

Cause: The CSS daemon aborted on the listed node with the listed return code

Action: Collect the CSS daemon logs from all nodes and any CSS daemon core files and contact Oracle Support

CRS-1603 CSSD on node %s shutdown by user.

Cause: The CSS daemon on the listed node was terminated by a user

Action: None

CRS-1604 CSSD voting file is offline: %s. Details in %s.

Cause: The listed voting file became unusable on the local node

Action: Verify that the file system containing the listed voting file is available on the local node

CRS-1605 CSSD voting file is online: %s. Details in %s.

Cause: The CSS daemon has detected a valid configured voting file

Action: None

CRS-1606 CSSD Insufficient voting files available [%s of %s]. Details in %s.

Cause: The number of voting files has decreased to a number of files that is insufficient.

Action: Locate previous 1604 messages and take action as indicated for message 1604

CRS-1607 CSSD evicting node %s. Details in %s.

Cause: The local node is evicting the indicated node

Action: Collect the CSS daemon logs from all nodes and any CSS daemon core files and contact Oracle Support

CRS-1608 CSSD Evicted by node %s. Details in %s.

Cause: The local node was evicted by the indicated node

Action: Collect the CSS daemon logs from all nodes and any CSS daemon core files and contact Oracle Support

CRS-1609 CSSD detected a network split. Details in %s.

Cause: Heartbeat messages between one or more nodes were not received and one or more nodes were evicted from the cluster to preserve data integrity.

Action: Verify all network connections between cluster nodes and repair any problematic connections. If there do not appear to be any network problems:

1. Collect the CSS daemon logs, system messages and any CSS daemon core files from all nodes

2. Contact Oracle Support

CRS-1610 node %s (%d) at 90%% heartbeat fatal, eviction in %d.%03d seconds

Cause: Did not receive heartbeat messages from the node. This could be due network problems or failure of listed node.

Action: Check if the private interconnect network used by cluster is functioning properly, including all the cables, network cards, switches/routers and so on between this node and listed node. Correct any problems discovered.

CRS-1611 node %s (%d) at 75%% heartbeat fatal, eviction in %d.%03d seconds

Cause: Did not receive heartbeat messages from the node. This could be due to network problems or failure of the listed node.

Action: Check if the private interconnect network used by cluster is functioning properly, including all the cables, network cards, switches/routers and so on between this node and listed node. Correct any problems discovered.

CRS-1612 node %s (%d) at 50%% heartbeat fatal, eviction in %d.%03d seconds

Cause: Did not receive heartbeat messages from the node. This could be due to network problems or failure of listed node.

Action: Check if the private interconnect network used by cluster is functioning properly, including all the cables, network cards, switches/routers and so on between this node and listed node. Correct any problems discovered.

CRS-1613 voting device hang at 90%% fatal, termination in %u ms, disk (%d/%s)

Cause: Voting device I/O has not completed for a long time. This could be due some error with the device voting file is on or in some element in the path of the I/O to the device.

Action: Verify if the device is working properly including all the I/O paths. Voting file listed will be considered inactive in the number of seconds specified. Failure of a majority of devices would result in node reboot.

CRS-1614 voting device hang at 75%% fatal, termination in %u ms, disk (%d/%s)

Cause: Voting device I/O has not completed for a long time. This could be due some error with the device voting file is on or in some element in the path of the I/O to the device.

Action: Verify if the device is working properly including all the I/O paths. Voting file listed will be considered inactive in the number of seconds specified. Failure of a majority of devices would result in node reboot.

CRS-1615 voting device hang at 50%% fatal, termination in %u ms, disk (%d/%s)

Cause: Voting device I/O has not completed for a long time. This could be due some error with the device voting file is on or in some element in the path of the I/O to the device.

Action: Verify if the device is working properly including all the I/O paths. Voting file listed will be considered inactive in the number of seconds specified. Failure of a majority of devices would result in node reboot.

CRS-1801 Cluster %s configured with nodes %s.

Cause: None

Action: None

CRS-1802 Node %s added to cluster.

Cause: None

Action: None

CRS-1803 Node %s deleted from cluster.

Cause: None

Action: None

CRS-1804 Node %s upgraded to version %s.

Cause: None

Action: None

CRS-1901 CRS service setting (%s) is changed from [%s] to [%s].

Cause: None

Action: None

CRS-2001 memory allocation error when initiating the connection failed to allocate memory for the connection with the target process

Cause: Failed to allocate memory for the connection with the target process.

Action: None.

CRS-2002 connection by user %s to %s refused

Cause: User command cannot connect to the target process.

Action: The user may not have sufficient privilege to connect.

CRS-2003 error %d encountered when connecting to %s

Cause: Connection to the target process failed.

Action: Examine whether the connection is made properly. Retry again at a later time if necessary.

CRS-2004 error %d encountered when sending messages to %s

Cause: User command cannot communicate with the target process properly.

Action: Retry again at a later time.

CRS-2005 timed out when waiting for response from %d

Cause: The target process does not return acknowledgment in time.

Action: Retry again at a later time.

CRS-2006 failed to get response from %d

Cause: The target process failed to return acknowledgement.

Action: Retry again at a later time.

CRS-2007 invalid component key name <%s> used

Cause: The given component key name could not be recognized.

Action: re-run the command with a valid component key name.

CRS-2008 invalid message type <%d> used

Cause: An unrecognized message type was sent.

Action: Retry with a valid command again.

CRS-2009 unable to get authentication for user %s

Cause: Current user was not authenticated for connection.

Action: Log in as another user and try again.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X