Oracle® Database

Administrator's Reference

10*g* Release 1 (10.1) for UNIX Systems: AIX-Based Systems, hp HP-UX, hp Tru64 UNIX, Linux, and Solaris Operating System (SPARC)

Part No. B10812-02

March 2004

ORACLE®

Oracle Database Administrator's Reference, 10*g* Release 1 (10.1) for UNIX Systems: AIX-Based Systems, hp HP-UX, hp Tru64 UNIX, Linux, and Solaris Operating System (SPARC)

Part No. B10812-02

Contents

4 Administering SQL*Plus

5 Configuring Oracle Net Services

6 Using Oracle Precompilers and the Oracle Call Interface

7 SQL*Loader and PL/SQL Demonstrations

8 Tuning for Oracle Database on UNIX

A Administering Oracle Database on AIX

B Administering Oracle Database on HP-UX

C Administering Oracle Database on Linux

D Administering Oracle Database on Tru64 UNIX

E Administering Oracle Database on Solaris

F Database Limits

Index

Send Us Your Comments

Oracle Database Administrator's Reference 10*g* Release 1 (10.1) for UNIX Systems: AIX-Based Systems, hp HP-UX, hp Tru64 UNIX, Linux, and Solaris Operating System (SPARC)

Part No. B10812-02

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find errors or have other suggestions for improvement, you can e-mail them to us at `osdwrite_us@oracle.com`. Please indicate the title and part number of the document and the chapter, section, and page number if available. Please indicate whether you would like a reply.

If you have problems with the software, please contact your local Oracle Support Services Center.

Preface

This guide provides information about administering and configuring Oracle Database 10*g* release 1 (10.1) on UNIX systems.

Audience

This guide is intended for anyone responsible for administering and configuring Oracle Database 10*g* release 1 (10.1) on a UNIX system. If you are configuring Oracle Real Application Clusters on a UNIX cluster, use this guide and the *Oracle Real Application Clusters Installation and Configuration Guide* for information about installation and configuration.

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. 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 additional information, visit the Oracle Accessibility Program Web site at:
`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, 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, JAWS 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.

Support for Hearing and Speech Impaired Customers

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 technical questions, call 1.800.446.2398
- For non-technical questions, call 1.800.464.2330

Terminology

The names for the UNIX operating systems have been shortened in this guide, as follows:

Operating System	Abbreviated Name
AIX-Based Systems	AIX
hp HP-UX PA-RISC (64-bit) hp HP-UX Itanium	HP-UX **Note:** Where there is a difference between the information for PA-RISC and Itanium systems, this is noted in the text.
hp Tru64 UNIX	Tru64 UNIX
Linux x86 Linux Itanium	Linux **Note:** Where there is a difference between the information for x86 and Itanium systems, this is noted in the text.
Solaris Operating System (SPARC)	Solaris

Typographic Conventions

The following typographic conventions are used in this guide:

Convention	Description
`monospace`	Monospace type indicates UNIX commands, directory names, usernames, pathnames, and filenames.
italics	Italic type indicates variables, including variable portions of filenames. It is also used for emphasis and for book titles.
UPPERCASE	Uppercase letters indicate Structured Query Language (SQL) reserved words, initialization parameters, and environment variables.

Command Syntax

UNIX command syntax appears in `monospace` font. The dollar character ($), number sign (#), or percent character (%) are UNIX command prompts. Do not enter them as part of the command. The following command syntax conventions are used in this guide:

Convention	Description

backslash \	A backslash is the UNIX command continuation character. It is used in command examples that are too long to fit on a single line. Enter the command as displayed (with a backslash) or enter it on a single line without a backslash:
	`dd if=/dev/rdsk/c0t1d0s6 of=/dev/rst0 bs=10b \`
	`count=10000`

braces { }	Braces indicate required items: `.DEFINE {macro1}`
brackets [ ]	Brackets indicate optional items: `cvtcrt` *`termname`* `[`*`outfile`*`]`
ellipses ...	Ellipses indicate an arbitrary number of similar items: `CHKVAL fieldname value1 value2 ... valueN`
italics	Italic type indicates a variable. Substitute a value for the variable: *`library_name`*
vertical line |	A vertical line indicates a choice within braces or brackets: `FILE` *`filesize`* `[K|M]`

Accessing Documentation

The documentation for Oracle Database 10*g* release 1 (10.1) for UNIX Systems includes platform-specific documentation and generic product documentation.

Platform-Specific Documentation

Platform-specific documentation includes information about installing and using Oracle products on particular platforms. The platform-specific documentation for this product is available in both Adobe portable document format (PDF) and HTML format on the product disc. To access the platform-specific documentation on disc:

1. Use a Web browser to open the `welcome.htm` file in the top-level directory of the disc.
2. For DVD-ROMs only, select the appropriate product link.
3. Select the **Documentation** tab.

If you prefer paper documentation, then open and print the PDF files.

Product Documentation

Product documentation includes information about configuring, using, or administering Oracle products on any platform. The product documentation for Oracle Database 10*g* products is available in both HTML and PDF formats in the following locations:

- On the Oracle Database 10*g* Documentation Library CD-ROM

  To access the documentation from the CD-ROM, use a Web browser to view the `index.htm` file in the top-level directory on the disc.

- In the `doc` subdirectory on the Oracle Database 10*g* DVD-ROM

  To access the documentation from the DVD-ROM, use a Web browser to view the `welcome.htm` file in the top-level directory on the disc, then select the Oracle Database 10*g* Documentation Library link.

- Online on the Oracle Technology Network (OTN) Web site:

  `http://otn.oracle.com/documentation`

Related Documentation

The platform-specific documentation for Oracle Database 10*g* products include the following manuals:

- Oracle Database:
  - *Oracle Database Release Notes* (platform-specific)
  - *Oracle Database Quick Installation Guide* (platform-specific)
  - *Oracle Database Installation Guide for UNIX Systems*
  - *Oracle Real Application Clusters Installation and Configuration Guide*
  - *Oracle Database Administrator's Reference for UNIX Systems*
  - *Oracle Procedural Gateway for APPC Installation and Configuration Guide for UNIX*
  - *Oracle Procedural Gateway for APPC User's Guide for UNIX*
  - *Oracle Procedural Gateway for APPC Messages Guide*
  - *Oracle Transparent Gateway for DRDA Installation and User's Guide for UNIX*
- Oracle Client:
  - *Oracle Database Client Quick Installation Guide* (platform-specific)
  - *Oracle Database Client Installation Guide for UNIX Systems*
- Oracle Database 10*g* Companion CD:
  - *Oracle Database Companion CD Installation Guide for UNIX Systems*
  - *Oracle Database Companion CD Quick Installation Guide* (platform-specific)

Refer to the Oracle Database release notes for your platform for important information that was not available when this book was released. The release notes for Oracle Database 10*g* are updated regularly. You can get the most-recent version from OTN:

`http://otn.oracle.com/documentation`

Third Party Software Notices

This program contains third party software from HP. The Oracle program license that accompanied this product determines your right to use the Oracle program, including the HP software. Notwithstanding anything to the contrary in the Oracle program license, the HP software is provided "AS IS" and without intellectual property indemnities, warranties, or support of any kind from Oracle or HP.

This program contains third party software from International Business Machines Corporation ("IBM"). The Oracle program license that accompanied this product determines your right to use the Oracle program, including the IBM software.

Notwithstanding anything to the contrary in the Oracle program license, the IBM software is provided "AS IS" and without intellectual property indemnities, warranties, or support of any kind from Oracle or IBM.

1

Administering Oracle on UNIX

This chapter provides information about administering Oracle Database on UNIX systems. It contains the following sections:

Overview

You must set Oracle Database environment variables, parameters, and user settings for Oracle Database to work. This chapter describes the various settings for Oracle Database on UNIX.

In Oracle Database files and programs, a question mark (?) represents the value of the ORACLE_HOME environment variable. For example, Oracle Database expands the question mark in the following SQL statement to the full path of the Oracle home directory:

```
SQL> ALTER TABLESPACE TEMP ADD DATAFILE '?/dbs/temp02.dbf' SIZE 200M
```

Similarly, the @ sign represents the ORACLE_SID environment variable. For example, to indicate a file belonging to the current instance, enter:

```
SQL> ALTER TABLESPACE tablespace_name ADD DATAFILE tempfile@.dbf
```

Environment Variables

This section describes the most commonly-used Oracle Database and UNIX environment variables. You must define some of these environment variables before installing Oracle Database. These environment variables are listed in the O*racle Database Installation Guide for UNIX Systems*.

To display the current value of an environment variable, use the `env` command. For example, to display the value of the ORACLE_SID environment variable, enter:

```
$ env | grep ORACLE_SID
```

To display the current value of all environment variables, use the `env` command as follows:

```
$ env | more
```

Oracle Database Environment Variables

Table 1–1 shows the syntax required by the environment variables used with Oracle Database, and provides examples of their values.

Table 1–1 Oracle Database Environment Variables on UNIX

Variable	Detail	Definition
NLS_LANG	Function	Specifies the language, territory, and character set of the client environment. The character set specified by NLS_LANG must match the character set of the terminal or terminal emulator. The character set specified by NLS_LANG can be different from the database character set, in which case Oracle automatically converts the character set. See the *Oracle Database Globalization Support Guide* for a list of values for this variable.
	Syntax	*language_territory.characterset*
	Example	`french_france.we8dec`
ORA_NLS10	Function	Specifies the directory where the language, territory, character set, and linguistic definition files are stored.
	Syntax	*directory_path*
	Example	`$ORACLE_HOME/nls/data`
ORA_TZFILE	Function	Specifies the full path and file name of the time zone file. You must set this environment variable if you want to use the small time zone file (`$ORACLE_HOME/oracore/zoneinfo/timezone.dat`) for data in the database. Oracle Database 10*g* uses the large time zone file by default (`$ORACLE_HOME/oracore/zoneinfo/timezlrg.dat`). This file contains information on more time zones than the small time zone file. All databases that share information must use the same time zone file. You must stop and restart the database if you change the value of this environment variable.
	Syntax	*directory_path*
	Example	`$ORACLE_HOME/oracore/zoneinfo/timezlrg.dat`
ORACLE_BASE	Function	Specifies the base of the Oracle directory structure for Optimal Flexible Architecture (OFA) compliant installations.
	Syntax	*directory_path*
	Example	`/u01/app/oracle`
ORACLE_HOME	Function	Specifies the directory containing the Oracle software.
	Syntax	*directory_path*
	Example	`$ORACLE_BASE/product/10.1.0/db_1`

Table 1–1 Oracle Database Environment Variables on UNIX (Cont.)

Variable	Detail	Definition
ORACLE_PATH	Function	Specifies the search path for files used by Oracle applications such as SQL*Plus. If the full path to the file is not specified, or if the file is not in the current directory, the Oracle application uses ORACLE_PATH to locate the file.
	Syntax	Colon-separated list of directories: *directory1:directory2:directory3*
	Example	`/u01/app/oracle/product/10.1.0/db_1/bin:.` **Note:** The period adds the current working directory to the search path.
ORACLE_SID	Function	Specifies the Oracle system identifier.
	Syntax	A string of numbers and letters that must begin with a letter. Oracle recommends a maximum of eight characters for system identifiers. For more information about this environment variable, see the *Oracle Database Installation Guide for UNIX Systems*.
	Example	SAL1
ORACLE_TRACE	Function	Enables the tracing of shell scripts during an installation. If it is set to T, many Oracle shell scripts use the `set -x` command, which prints commands and their arguments as they are run. If it is set to any other value, or no value, the scripts do not use the `set -x` command.
	Syntax	T or not T.
ORAENV_ASK	Function	Controls whether the `oraenv` or `coraenv` script prompts for the value of the ORACLE_SID environment variable. If it is set to NO, the scripts do not prompt for the value of the ORACLE_SID environment variable. If it is set to any other value, or no value, the scripts prompt for a value for the ORACLE_SID environment variable.
	Syntax	NO or not NO.
	Example	NO
SQLPATH	Function	Specifies the directory or list of directories that SQL*Plus searches for a `login.sql` file.
	Syntax	Colon-separated list of directories: *directory1:directory2:directory3*
	Example	`/home:/home/oracle:/u01/oracle`
TNS_ADMIN	Function	Specifies the directory containing the Oracle Net Services configuration files.
	Syntax	*directory_path*
	Example	`$ORACLE_HOME/network/admin`

Table 1–1 Oracle Database Environment Variables on UNIX (Cont.)

Variable	Detail	Definition
TWO_TASK	Function	Specifies the default connect identifier to use in the connect string. If this environment variable is set, you do not need to specify the connect identifier in the connect string. For example, if the TWO_TASK environment variable is set to `sales`, you can connect to a database using the `CONNECT` *`username/password`* command rather than the `CONNECT` *`username/password`*`@sales` command.
	Syntax	Any connect identifier.
	Range of Values	Any valid connect identifier that can be resolved using a naming method, such as a `tnsnames.ora` file or a directory server.
	Example	PRODDB_TCP

Note: To prevent conflicts, do not define environment variables with names that are identical to the names of Oracle Server processes, for example ARCH, PMON, and DBWR.

UNIX Environment Variables

Table 1–2 shows the syntax required by the UNIX environment variables used with Oracle Database and provides examples of their values.

Table 1–2 UNIX Environment Variables Used with Oracle Database

Variable	Detail	Definition
ADA_PATH (AIX only)	Function	Specifies the directory containing the Ada compiler.
	Syntax	*`directory_path`*
	Example	`/usr/lpp/powerada`
CLASSPATH	Function	Used with Java applications. The required setting for this variable depends on the Java application. See the product documentation for your Java application for more information.
	Syntax	Colon-separated list of directories or files: *`directory1:directory2:file1:file2`*
	Example	There is no default setting. CLASSPATH must include the following directories: `$ORACLE_HOME/JRE/lib:$ORACLE_HOME/jlib`

Table 1–2 UNIX Environment Variables Used with Oracle Database (Cont.)

Variable	Detail	Definition
DISPLAY	Function	Used by X-based tools. Specifies the display device used for input and output. See your X Window System documentation for information.
	Syntax	*hostname*:*server*[.*screen*] where *hostname* is the system name (either IP address or alias), *server* is the sequential code number for the server, and *screen* is the sequential code number for the screen. If you have a single monitor, use the value 0 for both server and screen (0.0). **Note:** If you have a single monitor, *screen* is optional.
	Example	135.287.222.12:0.0 bambi:0
HOME	Function	The user's home directory.
	Syntax	*directory_path*
	Example	/home/oracle
LANG or LANGUAGE	Function	Specifies the language and character set used by the operating system for messages and other output. See the operating system documentation for more information.
LD_OPTIONS	Function	Specifies the default linker options. See the ld man page for more information about this environment variable.
LPDEST (Solaris only)	Function	Specifies the name of the default printer.
	Syntax	*string*
	Example	docprinter
LD_LIBRARY_PATH (All platforms except AIX. On HP-UX, specifies the path for 64-bit shared libraries.)	Function	Specifies the list of directories that the shared library loader searches to locate shared object libraries at runtime. See the ld man page for information about this environment variable.
	Syntax	Colon-separated list of directories: *directory1:directory2:directory3*
	Example	/usr/dt/lib:$ORACLE_HOME/lib
LD_LIBRARY_PATH_64 (Solaris only)	Function	Specifies the list of directories that the shared library loader searches to locate specific 64-bit shared object libraries at runtime. See the ld man page for information about this environment variable.
	Syntax	Colon separated list of directories: *directory1:directory2:directory3*
	Example	/usr/dt/lib:$ORACLE_HOME/lib64

Table 1–2 UNIX Environment Variables Used with Oracle Database (Cont.)

Variable	Detail	Definition
LIBPATH (AIX only)	Function	Specifies the list of directories that the shared library loader searches to locate shared object libraries at runtime. See the `ld` man page for information about this environment variable.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/usr/dt/lib:$ORACLE_HOME/lib`
PATH	Function	Used by the shell to locate executable programs; must include the `$ORACLE_HOME/bin` directory.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/bin:/usr/bin:/usr/local/bin:/usr/bin/X11: $ORACLE_HOME/bin:$HOME/bin:.` **Note**: The period adds the current working directory to the search path.
PRINTER	Function	Specifies the name of the default printer.
	Syntax	*`string`*
	Example	`docprinter`
SHLIB_PATH (HP-UX 32-bit libraries only)	Function	Specifies the list of directories that the shared library loader searches to locate shared object libraries at runtime. See the `ld` man page for information about this environment variable.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/usr/dt/lib:$ORACLE_HOME/lib32`
TEMP, TMP, and TMPDIR	Function	Specify the default directories for temporary files; if set, tools that create temporary files create them in one of these directories.
	Syntax	*`directory_path`*
	Example	`/u02/oracle/tmp`
XENVIRONMENT	Function	Specifies a file containing X Window System resource definitions. See your X Window System documentation for more information.

Setting a Common Environment

This section describes how to set a common UNIX environment using the `oraenv` or `coraenv` scripts, depending on your default shell:

- For the Bourne, Bash, or Korn shell, use the `oraenv` command.
- For the C or tcsh shell, use the `coraenv` command.

oraenv and coraenv Script Files

The `oraenv` and `coraenv` scripts are created during installation. These scripts set environment variables based on the contents of the `oratab` file and provide:

- A central means of updating all user accounts with database changes
- A mechanism for switching between databases specified in the `oratab` file

You might find yourself frequently adding and removing databases from your development system or your users might be switching between several different Oracle databases installed on the same system. You can use the `oraenv` or `coraenv` script to ensure that user accounts are updated and to switch between databases.

Note: Do not call the `oraenv` or `coraenv` script from the Oracle software owner (typically `oracle`) user's shell startup script. Because these scripts prompt for values, they can prevent the `dbstart` script from starting a database automatically when the system starts.

The `oraenv` or `coraenv` script is usually called from the user's shell startup file (for example `.profile` or `.login`). It sets the ORACLE_SID and ORACLE_HOME environment variables and includes the `$ORACLE_HOME/bin` directory in the PATH environment variable setting. When switching between databases, users can run the `oraenv` or `coraenv` script to set these environment variables.

Note: To run one of these scripts, enter the appropriate command:

- For the `coraenv` script:

  ```
  % source /usr/local/bin/coraenv
  ```

- For the `oraenv` script:

  ```
  $ . /usr/local/bin/oraenv
  ```

Local bin Directory

The directory that contains the `oraenv`, `coraenv`, and `dbhome` scripts is called the local `bin` directory. All database users should have read access to this directory.

Include the path of the local `bin` directory in the users' PATH environment variable setting. When you run the `root.sh` script after installation, the script prompts you for the path of the local `bin` directory and automatically copies the `oraenv`, `coraenv`, and `dbhome` scripts to the directory that you specify. The default local `bin` directory is `/usr/local/bin`. If you do not run the `root.sh` script, you can manually copy the `oraenv` or `coraenv` and `dbhome` scripts from the `$ORACLE_HOME/bin` directory to the local `bin` directory.

Setting the System Time Zone

The TZ environment variable sets the time zone. It enables you to adjust the clock for daylight saving time changes or different time zones. The adjusted time is used to time-stamp files, produce the output of the `date` command, and obtain the current SYSDATE.

Oracle recommends that you do not change your personal TZ value. Using different values of TZ such as GMT+24 might change the date a transaction is recorded. This changed date affects Oracle applications that use SYSDATE. To avoid this problem, use sequence numbers to order a table instead of date columns.

Initialization Parameters

The following sections provide platform-specific information about Oracle Database initialization parameters:

- Maximum Value of the DB_BLOCK_SIZE Initialization Parameter
- Default Values for the ASM_DISKSTRING Initialization Parameter
- Maximum Value for ASYNC in LOG_ARCHIVE_DEST Initialization Parameter
- CLUSTER_INTERCONNECTS Initialization Parameter

Maximum Value of the DB_BLOCK_SIZE Initialization Parameter

The DB_BLOCK_SIZE initialization parameter specifies the standard block size for the database. This block size is used for the SYSTEM tablespace and by default in other tablespaces.

The maximum value to which you can set the DB_BLOCK_SIZE parameter differs on UNIX platforms, as listed in the following table:

Note: You cannot change the value of the DB_BLOCK_SIZE parameter after you create a database.

Platform	Maximum Value
Linux (x86)	16 KB
Other operating systems	32 KB

Default Values for the ASM_DISKSTRING Initialization Parameter

Table 1–3 lists the platform-specific default values for the Automatic Storage Management ASM_DISKSTRING initialization parameter.

Note: Only ASM instances support the ASM_DISKSTRING initialization parameter.

Table 1–3 Default Values for the ASM_DISKSTRING Initialization Parameter

Platform	Default Search String
AIX	`/dev/rhdisk*`
HP-UX	`/dev/rdsk/*`
Linux	`/dev/raw/*`
Solaris	`/dev/rdsk/*`
Tru64 UNIX	`/dev/rdisk/*`

Maximum Value for ASYNC in LOG_ARCHIVE_DEST Initialization Parameter

The maximum value that you can set for ASYNC in the LOG_ARCHIVE_DEST initialization parameter differs on UNIX platforms, as listed in the following table:

Platform	Maximum Value
HP-UX and Tru64 UNIX	51200
AIX, Solaris, and Linux	102400

CLUSTER_INTERCONNECTS Initialization Parameter

In a Real Application Clusters environment, you can use the CLUSTER_INTERCONNECTS initialization parameter to specify an alternative interconnect for the private network.

The CLUSTER_INTERCONNECTS parameter requires the IP address of the interconnect instead of the device name. It allows you to specify multiple IP addresses, separated by colons. Oracle Real Application Clusters network traffic is distributed between all of the specified IP addresses.

The CLUSTER_INTERCONNECTS parameter is useful only in an Oracle Real Application Clusters environment where UDP IPC is enabled. It enables users to specify an interconnect for all IPC traffic that includes Oracle Global Cache Service (GCS), Global Enqueue Service (GES), and Interprocessor Parallel Query (IPQ).

Overall cluster stability and performance might improve when you force Oracle GCS, GES, and IPQ over a different interconnect by setting the CLUSTER_INTERCONNECTS parameter. For example, to use the network interface whose IP address is 129.34.137.212 for all GCS, GES, and IPQ IPC traffic, set the CLUSTER_INTERCONNECTS parameter as follows:

```
CLUSTER_INTERCONNECTS=129.34.137.212
```

Use the `ifconfig` or `netstat` commands to display the IP address of a device. This command provides a map between device names and IP addresses. For example, to determine the IP address of a device on Tru64 UNIX, enter the following command as the `root` user:

```
# /usr/sbin/ifconfig -a
fta0: flags=c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX>
      inet 129.34.137.212 netmask fffffc00 broadcast 129.34.139.255 ipmtu 1500

lo0:  flags=100c89<UP,LOOPBACK,NOARP,MULTICAST,SIMPLEX,NOCHECKSUM>
      inet 127.0.0.1 netmask ff000000 ipmtu 4096
```

```
ics0:  flags=1100063<UP,BROADCAST,NOTRAILERS,RUNNING,NOCHECKSUM,CLUIF>
      inet 10.0.0.1 netmask ffffff00 broadcast 10.0.0.255 ipmtu 7000

sl0:  flags=10<POINTOPOINT>

tun0: flags=80<NOARP>
```

In the preceding example, the interface `fta0:` has an IP address of 129.34.137.212 and the interface `ics0:` has an IP address of 10.0.0.1.

Note the following important points when using the CLUSTER_INTERCONNECTS initialization parameter:

- The IP addresses specified for the different instances of the same database on different nodes should belong to network adaptors that connect to the same network. If you do not follow this rule, internode traffic might pass through bridges and routers or there might not be a path between the two nodes at all.
- Specify the CLUSTER_INTERCONNECTS parameter in the parameter file, setting a different value for each database instance.
- If you specify multiple IP addresses for this parameter, list them in the same order for all instances of the same database. For example on Tru64 UNIX, if the parameter for instance 1 on node 1 lists the IP addresses of the `alt0:`, `fta0:`, and `ics0:` devices in that order, the parameter for instance 2 on node 2 should list the IP addresses of the equivalent network adaptors in the same order.
- If the interconnect IP address specified is incorrect or does not exist on the system, Oracle Database uses the default cluster interconnect device. On Tru64 UNIX, for example, the default device is `ics0:`.

Failover and Failback and CLUSTER_INTERCONNECTS

Some operating systems support runtime failover and failback. However, if you use the CLUSTER_INTERCONNECTS initialization parameter, failover and failback are disabled.

> **See Also:** For more information about runtime failover and failback on AIX systems, refer to Appendix A, "Administering Oracle Database on AIX".

Operating System Accounts and Groups

Special operating system accounts and groups are required by Oracle Database, as follows:

- Oracle software owner account
- OSDBA, OSOPER, and Oracle Inventory groups

Oracle Software Owner Account

The Oracle software owner account, usually named `oracle`, is the account that you use to install the Oracle software. You can use different Oracle software owner accounts to install the software in separate Oracle home directories. However, for each Oracle home directory, you must use the same account that installed the software for all subsequent maintenance tasks on that Oracle home directory.

Oracle recommends that the Oracle software owner has the Oracle Inventory group as its primary group and the OSDBA group as its secondary group.

OSDBA, OSOPER, and Oracle Inventory Groups

Table 1–4 describes the special UNIX groups required by Oracle Database.

Table 1–4 UNIX Groups

Group	Typical Name	Description
OSDBA	`dba`	Operating system accounts that are members of the OSDBA group have special database privileges. Members of this group can connect to the database using the SYSDBA privilege. The Oracle software owner is the only required member of this group. You can add other accounts as required.
OSOPER	`oper`	The OSOPER group is an optional group. Operating system accounts that are members of the OSOPER group have special database privileges. Members of this group can connect to the database using the SYSOPER privilege.
Oracle Inventory	`oinstall`	All users installing Oracle software on a UNIX system must belong to the same UNIX group, called the Oracle Inventory group. This group must be the primary group of the Oracle software owner during installations. After the installation, this group owns all of the Oracle files installed on the system.

> **See Also:** For more information about the OSDBA group and SYSDBA privileges, and the OSOPER group and SYSOPER privileges, see the *Oracle Database Administrator's Guide* and the *Oracle Database Installation Guide for UNIX Systems*.

Oracle Database uses several features of the UNIX operating system to provide a secure environment for users. These features include file ownership, group accounts, and the ability of a program to change its user ID upon execution.

The two-task architecture of Oracle Database improves security by dividing work (and address space) between the user program and the `oracle` program. All database access is achieved through the shadow process and special authorizations in the `oracle` program.

> **See Also:** For more information about security issues, see the *Oracle Database Administrator's Guide*.

Groups and Security

Oracle programs are divided into two sets for security purposes: those executable by all (`other` in UNIX terms), and those executable by DBAs only. Oracle recommends that you take the following approach to security:

- The primary group for the `oracle` account should be the `oinstall` group.
- The `oracle` account must have the `dba` group as a secondary group.
- Although any user account that requires the SYSDBA privilege can belong to the `dba` group, the only user accounts that should belong to the `oinstall` group are the Oracle software owner accounts, for example, `oracle`.

External Authentication

If you choose to use external authentication, you must use the value of the OS_AUTHENT_PREFIX initialization parameter as a prefix for Oracle user names. If you do not explicitly set this parameter, the default value on UNIX is `ops$`, which is case sensitive.

To use the same user names for both operating system and Oracle authentication, set this initialization parameter to a null string, as follows:

```
OS_AUTHENT_PREFIX=""
```

See Also: For more information about external authentication, see the *Oracle Database Administrator's Guide*.

Running the orapwd Utility

You can use a password file to identify users that can use the SYSDBA and SYSOPER privileges when connecting to the database. If you use the Database Configuration Assistant (DBCA) to create a database, it creates a password file for the new database. If you create the database manually, create the password file for it as follows:

1. Log in as the Oracle software owner.
2. Use the `orapwd` utility to create the password file, as follows:

```
$ $ORACLE_HOME/bin/orapwd file=filename password=password entries=max_users
```

The following table describes the values that you must specify in this command:

Value	Description
filename	The name of the file where password information is written. The name of the file must be `orapwsid` and you must supply the full path name. Its contents are encrypted and are not user-readable. The password file is typically created in the `$ORACLE_HOME/dbs` directory.
password	The password for the SYS user. If you use an ALTER USER statement to change the password for the SYS user after you connect to the database, both the password stored in the data dictionary and the password stored in the password file are updated. This parameter is mandatory.
max_users	Sets the maximum number of entries allowed in the password file. This is the maximum number of distinct users allowed to connect to the database simultaneously with either the SYSDBA or the SYSOPER privilege.

See Also: For more information about using the `orapwd` utility, see the *Oracle Database Administrator's Guide*.

The orapwd Utility and Oracle Real Application Clusters

If you are using the file system for database file storage and if the Real Application Clusters Oracle home directory is not on a cluster file system, you can set the

password file to be a symbolic link to a shared file. If you choose to implement this, changes that you make to the file through SQL are shared across all instances. Otherwise, the file is updated only on the node with the instance that issues the SQL, and the file must be copied to all other nodes that run instances of the database. For this reason, Oracle recommends that you set `$ORACLE_HOME/dbs/orapw`*`sid`* to be a symbolic link to a shared file.

Password Management

When using the Database Configuration Assistant to create a database, users must change the SYS and SYSTEM account passwords. You cannot use the default CHANGE_ON_INSTALL and MANAGER passwords.

For security reasons, the Database Configuration Assistant locks most Oracle user accounts after it creates the database. It does not lock the SYS or SYSTEM accounts. You must unlock any locked accounts and change their passwords before logging into them.

To change the passwords in the DBCA, click the Password Management button in the Database Configuration Assistant Summary window.

Alternatively, use SQL*Plus to connect to the database as SYS and enter the following command to unlock an account and reset its password:

```
SQL> ALTER USER username IDENTIFIED BY passwd ACCOUNT UNLOCK;
```

Creating Additional UNIX Accounts

If necessary, create additional UNIX accounts. Users must be members of the OSDBA or OSOPER groups to connect to the database with administrator privileges.

Configuring the Accounts of Oracle Users

Update the startup files of the `oracle` user and the UNIX accounts of Oracle users, specifying the appropriate environment variables in the environment file.

For the Bourne, Bash or Korn shell, add the environment variables to the `.profile` file, or on Red Hat Enterprise Linux, the `.bash_profile` file.

For the C or tcsh shell, add the environment variables to the `.login` file.

Note: You can use the `oraenv` or `coraenv` script to ensure that Oracle user accounts are updated.

Using Raw Devices

The following sections provide information about using raw devices (raw partitions or raw volumes).

Note: For additional raw device tuning information, see the following appendices:

- Appendix A, "Administering Oracle Database on AIX"
- Appendix C, "Administering Oracle Database on Linux"
- Appendix D, "Administering Oracle Database on Tru64 UNIX"

Guidelines for Using Raw Devices

Raw devices (raw partitions or raw volumes) have the following potential disadvantages when used on UNIX:

- Raw devices might not solve problems with file size writing limits.

  Note: To display current file size limits, enter one of the following commands:

  - Bourne, Bash, or Korn shell:

    ```
    $ ulimit -a
    ```

  - C or tcsh shell:

    ```
    % limit
    ```

- Small client systems might not be able to use sufficiently large partitions or volumes for the raw devices.
- If a particular disk drive has intense I/O activity and performance would benefit from movement of an Oracle data file to another drive, it is likely that no acceptably sized partition or volume exists on a drive with less I/O activity. It

might not be possible to move files to other disk drives if you are using raw devices.

- Raw devices are more difficult to administer than data files stored on a file system or in an Automatic Storage Management (ASM) disk group.

Consider the following issues when deciding whether to use raw devices:

- Oracle Real Application Clusters installation

  Each instance of Oracle Real Application Clusters (RAC) has its own log files. Therefore, in addition to the devices required for the tablespaces and control files, each instance requires a minimum of two partitions for the log files. All of the files must be on disks that can be shared by all nodes of a cluster.

- Raw disk partition availability

  Use raw partitions for Oracle files only if you have at least as many raw disk partitions as Oracle data files. If disk space is a consideration and the raw disk partitions are already created, match data file size to partition size as closely as possible to avoid wasting space.

  You must also consider the performance implications of using all of the disk space on a few disks as opposed to using less space on more disks.

- Logical volume manager

  Logical volume managers manage disk space at a logical level and hide some of the complexity of raw devices. With logical volumes, you can create logical disks based on raw partition availability. The logical volume manager controls fixed-disk resources by:

  – Mapping data between logical and physical storage

  – Allowing data to span multiple disks and to be discontiguous, replicated, and dynamically expanded

  For RAC, you can use logical volumes for drives associated with a single UNIX system, as well as those that can be shared with more than one system of a UNIX cluster. Shared drives allow for all files associated with a RAC database to be placed on these shared logical volumes.

- Dynamic performance tuning

  To optimize disk performance, you can move files from disk drives with high activity to disk drives with less activity. Most hardware vendors who provide the logical disk facility also provide a graphical user interface that you can use for tuning.

- Mirroring and online disk replacement

  You can mirror logical volumes to protect against loss of data. If one copy of a mirror fails, dynamic resynchronization is possible. Some vendors also provide the ability to replace drives online in conjunction with the mirroring facility.

Raw Device Setup

Keep the following items in mind when creating raw devices:

- When creating the devices, ensure that the owner is the Oracle software owner user (`oracle`) and the group is the OSDBA group (`dba`).
- The size of an Oracle data file created in a raw partition must be at least two Oracle block sizes smaller than the size of the raw partition.

> **See Also:** Fore more information about creating raw devices, see your operating system documentation.

Raw Device Datafiles on AIX or Tru64 UNIX Systems

On AIX and Tru64 UNIX systems, datafiles on raw logical volumes may have offsets for the first block of the Oracle data. This offset is required by the logical volume manager.

You can use the $ORACLE_HOME/bin/offset utility to determine the offset value, for example, if you want to transfer the datafile to a different device.

Using Trace and Alert Files

This section describes the trace (or dump) and alert files that Oracle Database creates to help you diagnose and resolve operating problems.

Trace Files

Each server and background process writes to a trace file. When a process detects an internal error, it writes information about the error to its trace file. The file name format of a trace file is *sid_processname_unixpid*.trc, where:

- *sid* is the instance system identifier
- *processname* is a three or four-character abbreviated process name identifying the Oracle Database process that generated the file (for example, pmon, dbwr, ora, or reco)
- *unixpid* is the UNIX process ID number

A sample trace file name is
$ORACLE_BASE/admin/TEST/bdump/test_lgwr_1237.trc

All trace files for background processes are written to the destination directory specified by the BACKGROUND_DUMP_ DEST initialization parameter. If you do not set this initialization parameter, the default directory is $ORACLE_HOME/rdbms/log.

All trace files for user processes are written to the destination directory specified by the USER_DUMP_DEST initialization parameter. If you do not set the USER_DUMP_DEST initialization parameter, the default directory is $ORACLE_HOME/rdbms/log. Set the MAX_DUMP_FILE initialization parameter to at least 5000 to ensure that the trace file is large enough to store error information.

Alert Files

The alert_*sid*.log file stores information about significant database events and messages. Events that affect the database instance or database are recorded in this file. This file is associated with a database and is located in the directory specified by the BACKGROUND_DUMP_DEST initialization parameter. If you do not set this initialization parameter, the default directory is $ORACLE_HOME/rdbms/log.

2

Starting and Stopping Oracle Software

This chapter describes how to identify Oracle Database processes running on UNIX systems, and provides basic information about how to stop and restart them. For more information about administering the products, see the appropriate product-specific documentation. This chapter contains information about the following:

- Stopping and Starting Oracle Processes
- Automating Startup and Shutdown

Stopping and Starting Oracle Processes

This section describes how to stop and start the processes for the following Oracle products:

- Database or Automatic Storage Management Instances
- Oracle Net Listener
- iSQL*Plus
- Oracle Ultra Search
- Oracle Enterprise Manager Database Control
- Oracle Management Agent

Database or Automatic Storage Management Instances

This section describes how to stop and start database or Automatic Storage Management (ASM) instances.

Stopping Database or ASM Instances

To stop an Oracle database or ASM instance, follow these steps:

1. If necessary, to identify the SID and Oracle home directory for the instance that you want to shut down, enter the following command:

   - For Solaris:

     ```
     $ cat /var/opt/oracle/oratab
     ```

   - For other operating systems:

     ```
     $ cat /etc/oratab
     ```

   The `oratab` file contains lines similar to the following, which identify the SID and corresponding Oracle home directory for each database or ASM instance on the system:

   sid:*oracle_home_directory*:[Y|N]

   Note: Oracle recommends that you use the plus sign (+) as the first character in the SID of ASM instances.

Caution: Do not stop an ASM instance until you have stopped all Oracle database instances that use that ASM instance to manage their storage.

2. Depending on your default shell, run the `oraenv` or `coraenv` script to set the environment variables for the instance that you want to shut down:

   - For the Bourne, Bash or Korn shell:

     ```
     $ . /usr/local/bin/oraenv
     ```

   - For the C shell:

     ```
     % source /usr/local/bin/coraenv
     ```

   When prompted, specify the SID for the instance.

3. Enter the following commands to shut down the instance:

   ```
   $ sqlplus /nolog
   SQL> CONNECT SYS/sys_password as SYSDBA
   SQL> SHUTDOWN NORMAL
   ```

   After the instance shuts down, you can exit from SQL*Plus.

Restarting Database or ASM Instances

To restart an Oracle database or Automatic Storage Management instance, follow these steps:

1. If necessary, repeat steps 1 and 2 in the previous subsection to set the ORACLE_SID and ORACLE_HOME environment variables to identify the SID and Oracle home directory for the instance that you want to start.

   Caution: If the database instance uses ASM for storage management, you must start the ASM instance before you start the database instance.

2. Enter the following commands to start the instance:

```
$ sqlplus /nolog
SQL> CONNECT SYS/sys_password as SYSDBA
SQL> STARTUP
```

After the instance starts, you can exit from SQL*Plus.

Oracle Net Listener

This section describes how to stop and start Oracle Net listener.

Stopping Oracle Net Listener

To stop an Oracle Net listener, follow these steps:

1. Enter the following command to determine the listener name and Oracle home directory for the Oracle Net listener that you want to stop:

```
$ ps -ef | grep tnslsnr
```

This command displays a list of the Oracle Net listeners running on the system, similar to the following, where *listenername1* and *listenername2* are the names of the listeners:

```
94248 ?? I 0:00.18 oracle_home1/bin/tnslsnr listenername1 -inherit
94248 ?? I 0:00.18 oracle_home2/bin/tnslsnr listenername2 -inherit
```

2. Set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for the listener that you want to stop:

   - For the Bourne, Bash, or Korn shell:

```
$ ORACLE_HOME=oracle_home1
$ export ORACLE_HOME
```

   - For the C or tcsh shell:

```
% setenv ORACLE_HOME oracle_home1
```

3. Enter the following command to stop the Oracle Net listener:

```
$ $ORACLE_HOME/bin/lsnrctl stop listenername
```

Note: If the name of the listener is the default name LISTENER, you do not have to specify the name in this command.

Restarting Oracle Net Listener

To start an Oracle Net listener, follow these steps:

1. If necessary, set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for the listener that you want to start:
   - For the Bourne, Bash, or Korn shell:

     ```
     $ ORACLE_HOME=oracle_home1
     $ export ORACLE_HOME
     ```

   - For the C or tcsh shell:

     ```
     % setenv ORACLE_HOME oracle_home1
     ```

2. Enter the following command to restart the Oracle Net listener:

   ```
   $ $ORACLE_HOME/bin/lsnrctl start [listenername]
   ```

   You must specify the listener name only if it is different from the default listener name LISTENER. To determine the listener name, enter the following command to view the `listener.ora` file:

   ```
   $ more $ORACLE_HOME/network/admin/listener.ora
   ```

*i*SQL*Plus

This section describes how to stop and start *i*SQL*Plus.

Stopping *i*SQL*Plus

To stop *i*SQL*Plus, follow these steps:

1. Set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for *i*SQL*Plus:
   - For the Bourne, Bash, or Korn shell:

     ```
     $ ORACLE_HOME=oracle_home
     $ export ORACLE_HOME
     ```

- For the C or tcsh shell:

```
% setenv ORACLE_HOME oracle_home
```

2. Enter the following command to stop *i*SQL*Plus:

```
$ $ORACLE_HOME/bin/isqlplusctl stop
```

Starting *i*SQL*Plus

To start *i*SQL*Plus, follow these steps:

1. If necessary, set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for the *i*SQL*Plus instance that you want to start:

   - For the Bourne, Bash, or Korn shell:

```
$ ORACLE_HOME=oracle_home
$ export ORACLE_HOME
```

   - For the C or tcsh shell:

```
% setenv ORACLE_HOME oracle_home
```

2. Enter the following command to start *i*SQL*Plus:

```
$ $ORACLE_HOME/bin/isqlplusctl start
```

Oracle Ultra Search

This section describes how to stop and start Oracle Ultra Search.

Stopping Oracle Ultra Search

To stop Oracle Ultra Search, follow these steps:

1. If necessary, set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for Oracle Ultra Search:

   - For the Bourne, Bash, or Korn shell:

```
$ ORACLE_HOME=oracle_home
$ export ORACLE_HOME
```

- For the C or tcsh shell:

```
% setenv ORACLE_HOME oracle_home
```

2. Enter the following command to stop Oracle Ultra Search:

```
$ $ORACLE_HOME/bin/searchctl stop
```

Starting Oracle Ultra Search

To start Oracle Ultra Search, follow these steps:

1. If necessary, set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for Oracle Ultra Search:

   - For the Bourne, Bash, or Korn shell:

```
$ ORACLE_HOME=oracle_home
$ export ORACLE_HOME
```

   - For the C or tcsh shell:

```
% setenv ORACLE_HOME oracle_home
```

2. Enter the following command to start Oracle Ultra Search:

```
$ $ORACLE_HOME/bin/searchctl start
```

Oracle Enterprise Manager Database Control

This section describes how to stop and start Oracle Enterprise Manager Database Control (Database Control).

Stopping Database Control

To stop Database Control, follow these steps:

1. Depending on your default shell, run the `oraenv` or `coraenv` script to set the environment for the database managed by the Database Control that you want to stop:

   - For the `coraenv` script:

```
% source /usr/local/bin/coraenv
```

- For the `oraenv` script:

  ```
  $ . /usr/local/bin/oraenv
  ```

2. Enter the following command to stop the Database Control:

   ```
   $ $ORACLE_HOME/bin/emctl stop dbconsole
   ```

Starting Database Control

To start Database Control, follow these steps:

1. If necessary, set the ORACLE_SID and ORACLE_HOME environment variables to identify the SID and Oracle home directory for the database control that you want to start:

   - For the Bourne, Bash, or Korn shell:

     ```
     $ ORACLE_HOME=oracle_home
     $ ORACLE_SID=sid
     $ export ORACLE_HOME ORACLE_SID
     ```

   - For the C or tcsh shell:

     ```
     % setenv ORACLE_HOME oracle_home
     % setenv ORACLE_SID sid
     ```

2. Enter the following command to start the Database Control:

   ```
   $ $ORACLE_HOME/bin/emctl start dbconsole
   ```

Oracle Management Agent

If you are using Oracle Enterprise Manager Grid Control to manage multiple Oracle products from a central location, you must have an Oracle Management Agent installed on each host system. Typically, the Oracle Management Agent is installed in its own Oracle home directory.

This section describes how to stop and start Oracle Management Agent.

Stopping Oracle Management Agent

To stop Oracle Management Agent, follow these steps:

1. If necessary, enter the following command to determine the Oracle home directory for Oracle Management Agent:

   ```
   $ ps -ef | grep emagent
   ```

   This command displays information about the Oracle Management Agent processes, similar to the following:

   ```
   94248 ?? I 0:00.18 oracle_home/agent/bin/emagent ...
   ```

2. Set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for the Oracle Management Agent:

   - For the Bourne, Bash, or Korn shell:

     ```
     $ ORACLE_HOME=oracle_home
     $ export ORACLE_HOME
     ```

   - For the C or tcsh shell:

     ```
     % setenv ORACLE_HOME oracle_home
     ```

3. Enter the following command to stop Oracle Management Agent:

   ```
   $ $ORACLE_HOME/agent/bin/emctl stop agent
   ```

Starting Oracle Management Agent

To start Oracle Management Agent, follow these steps:

1. If necessary, set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for Oracle Management Agent:

   - For the Bourne, Bash, or Korn shell:

     ```
     $ ORACLE_HOME=oracle_home
     $ export ORACLE_HOME
     ```

   - For the C or tcsh shell:

     ```
     % setenv ORACLE_HOME oracle_home
     ```

2. Enter the following command to start Oracle Management Agent:

   ```
   $ $ORACLE_HOME/agent/bin/emctl start agent
   ```

Automating Startup and Shutdown

> **Note:** The procedure described in this section applies to single-instance databases only.

Oracle recommends that you configure your system to automatically start Oracle databases when the system starts up, and to automatically shut them down when the system shuts down. Automating database startup and shutdown guards against incorrect database shutdown.

To automate database startup and shutdown, use the `dbstart` and `dbshut` scripts, located in the `$ORACLE_HOME/bin` directory. The scripts refer to the same entries in the `oratab` file, and so must apply to the same set of databases. You cannot, for example, have the `dbstart` script automatically start `sid1`, `sid2`, and `sid3`, and have the `dbshut` script shut down only `sid1`. However, you can specify that the `dbshut` script shuts down a set of databases while the `dbstart` script is not used at all. To do this, include a `dbshut` entry in the system shutdown file, but omit the `dbstart` entry from the system startup files.

> **See Also:** For more information about system startup and shutdown procedures, see the `init` command in your UNIX system documentation.

The following section describes how to automate startup and shutdown of Oracle Database.

Automating Database Startup and Shutdown

Use the `dbstart` and `dbshut` scripts to automate database startup and shutdown. Perform the following tasks to set up these scripts so that they are called at system startup and shutdown. Perform these steps for every new database that you want to configure for automated startup and shutdown:

1. Log in as the `root` user.
2. Edit the `oratab` file for your platform:
   - Solaris:

     ```
     /var/opt/oracle/oratab
     ```

- Other operating systems:

  `/etc/oratab`

Database entries in the `oratab` file appear in the following format:

`SID``:ORACLE_HOME:{Y|N}`

In this example, Y or N specifies whether you want the scripts to start up or shut down the database. For each database that you want to start up and shut down, find the instance identifier (SID) for that database, identified by the *`SID`* in the first field, and change the last field for each to Y.

3. Change directory to one of the following directories, depending on your platform:

Platform	Initialization File Directory
AIX	`/etc`
Linux and Solaris	`/etc/init.d`
HP-UX and Tru64 UNIX	`/sbin/init.d`

4. Create a file called `dbora` with the following contents. Change the value specified for the ORACLE_HOME environment variable to an Oracle home directory for your installation, and change the value of the ORACLE environment variable to the username of the owner of the database installed in the Oracle home directory (typically `oracle`):

```
#! /bin/sh  -x
#
# Change the value of ORACLE_HOME to specify the correct Oracle home
# directory for you installation

ORACLE_HOME=/u01/app/oracle/product/10.1.0/db_1
#
# change the value of ORACLE to the login name of the
# oracle owner at your site
#

ORACLE=oracle

PATH=${PATH}:$ORACLE_HOME/bin
HOST=`hostname`
```

```
PLATFORM=`uname`
export ORACLE_HOME PATH
#
if [ ! "$2" = "ORA_DB" ] ; then
   if [ "$PLATFORM" = "HP-UX" ] ; then
      remsh $HOST -l $ORACLE -n "$0 $1 ORA_DB"
      exit
   else
      rsh $HOST -l $ORACLE  $0 $1 ORA_DB
      exit
   fi
fi
#
LOG=$ORACLE_HOME/startup.log
touch $LOG
chmod a+r $LOG
#

case $1 in
'start')
        echo "$0: starting up" >> $LOG
        date >> $LOG
        # Start Oracle Net
        if [ -f $ORACLE_HOME/bin/tnslsnr ] ; then
             echo "starting Oracle Net Listener"
             $ORACLE_HOME/bin/lsnrctl start >> $LOG 2>&1 &
        fi
        echo "Starting Oracle databases"
        $ORACLE_HOME/bin/dbstart >> $LOG 2>&1 &
        ;;
'stop')
        echo "$0: shutting down" >> $LOG
        date >> $LOG
        # Stop Oracle Net
        if [ -f $ORACLE_HOME/bin/tnslsnr ] ; then
             echo "stopping Oracle Net Listener"
             $ORACLE_HOME/bin/lsnrctl stop >> $LOG 2>&1 &
        fi
        echo "stopping Oracle databases"
        $ORACLE_HOME/bin/dbshut >> $LOG 2>&1 &
        ;;
```

```
*)
        echo "usage: $0 {start|stop}"
        exit
        ;;
esac
#
exit
```

Note: This script works only if a password has not been set for the Oracle Net listener. If a password has been set for the listener, this script cannot stop the listener. Also, if the listener name is not the default name LISTENER, you must specify the listener name in the stop and start commands:

```
$ORACLE_HOME/bin/lsnrctl {start|stop} listener_name
```

---

5. Change the group of the `dbora` file to the OSDBA group (typically `dba`) and set the permissions to 750:

```
# chgrp dba dbora
# chmod 750 dbora
```

6. Create symbolic links to the `dbora` script in the appropriate run-level script directories, as follows:

Platform	Symbolic Links Commands
AIX	`# ln -s /etc/dbora /etc/rc.d/rc2.d/S99dbora` `# ln -s /etc/dbora /etc/rc.d/rc2.d/K01dbora`
HP-UX	`# ln -s /sbin/init.d/dbora /sbin/rc3.d/S990dbora` `# ln -s /sbin/init.d/dbora /sbin/rc3.d/K001dbora`

Linux	`# ln -s /etc/init.d/dbora /etc/rc.d/rc3.d/K01dbora`
	`# ln -s /etc/init.d/dbora /etc/rc.d/rc3.d/S99dbora`
	`# ln -s /etc/init.d/dbora /etc/rc.d/rc5.d/K01dbora`
	`# ln -s /etc/init.d/dbora /etc/rc.d/rc5.d/S99dbora`

Solaris	`# ln -s /etc/init.d/dbora /etc/rc3.d/K01dbora` `# ln -s /etc/init.d/dbora /etc/rc3.d/S99dbora`
Tru64 UNIX	`# ln -s /sbin/init.d/dbora /sbin/rc3.d/S99dbora` `# ln -s /sbin/init.d/dbora /sbin/rc3.d/K01dbora`

3

Configuring Oracle Products on UNIX

This chapter describes how to administer Oracle Database and Oracle software on UNIX systems. It contains the following sections:

- Configuring the Database for Additional Oracle Products
- Using Configuration Assistants as Standalone Tools
- Relinking Executables

Configuring the Database for Additional Oracle Products

If you install additional Oracle products after the initial installation, use the Database Configuration Assistant to configure your database for the new products, as follows.

1. Start the database, if necessary.
2. Enter the following command to start the Database Configuration Assistant:

   ```
   $ $ORACLE_HOME/bin/dbca
   ```

3. Select **Configure Database Options**.
4. From the list of available databases, select the database that you want to configure.
5. Choose the products that you want to enable from the list, then click **Finish**.

Using Configuration Assistants as Standalone Tools

Configuration Assistants are usually run during an installation session, but you can also run them in standalone mode. As with Oracle Universal Installer, you can run each of the assistants non-interactively using a response file. This section describes how to use the following Oracle configuration assistants:

- Using Oracle Net Configuration Assistant
- Using Database Upgrade Assistant
- Using Database Configuration Assistant
- Configuring New or Upgraded Databases

Using Oracle Net Configuration Assistant

When Oracle Net Server or Oracle Net Client is installed, Oracle Universal Installer automatically launches Oracle Net Configuration Assistant.

If you choose to perform a separate Oracle Client installation, then Oracle Net Configuration Assistant automatically creates a configuration that is consistent with the selections made during the installation. The Installer automatically runs the Oracle Net Configuration Assistant to set up a net service name in the local naming file located in the `$ORACLE_HOME/network/admin` directory of the client installation.

After installation is complete, you can use the Oracle Net Configuration Assistant to create a more detailed configuration, by entering the following command:

```
$ $ORACLE_HOME/bin/netca
```

Note: When you use the Database Configuration Assistant to create a database, it automatically updates the network configuration files to include information for the new database.

Using Database Upgrade Assistant

During an Oracle Database installation, the installer enables you to upgrade a database from an earlier release to the current release. However, if you choose not to upgrade a database during installation or if there is more than one database that you want to upgrade, you can run the Database Upgrade Assistant after the installation.

If you installed Oracle Database software and chose not to upgrade the database during the installation, then you must upgrade the database before mounting it.

To start the Database Upgrade Assistant, enter the following command:

```
$ $ORACLE_HOME/bin/dbua
```

For information about the command line options available with the Database Upgrade Assistant, use the `-help` or `-h` command line arguments, as follows:

```
$ $ORACLE_HOME/bin/dbua -help
```

See Also: For more information about upgrades, see the *Oracle Database Installation Guide for UNIX Systems* and the *Oracle Database Upgrade Guide*.

Using Database Configuration Assistant

You can use the Database Configuration Assistant to:

- Create a default or customized database
- Configure an existing database to use Oracle products
- Create Automatic Storage Management disk groups
- Generate a set of shell and SQL scripts that you can inspect, modify, and run at a later time to create a database

To start the Database Configuration Assistant, enter the following command:

```
$ $ORACLE_HOME/bin/dbca
```

For information about the command line options available with the Database Configuration Assistant, use the `-help` or `-h` command line arguments, as follows:

```
$ $ORACLE_HOME/bin/dbca -help
```

Configuring New or Upgraded Databases

Oracle recommends that you run the `utlrp.sql` script after creating or upgrading a database. This script recompiles all PL/SQL modules that might be in an invalid state, including packages, procedures, and types. This is an optional step but Oracle recommends that you do it when you create the database and not at a later date.

To run the `utlrp.sql` script, follow these steps:

1. Switch user to `oracle`.
2. Use the `oraenv` or `coraenv` script to set the environment for the database where you want to run the `utlrp.sql` script:
   - Bourne, Bash, or Korn shell:

     ```
     $ . /usr/local/bin/oraenv
     ```

   - C or tcsh shell:

     ```
     % source /usr/local/bin/coraenv
     ```

   When prompted, specify the SID for the database.
3. Start SQL*Plus, as follows:

   ```
   $ sqlplus "/ AS SYSDBA"
   ```

4. If necessary, start the database:

   ```
   SQL> STARTUP
   ```

5. Run the `utlrp.sql` script:

   ```
   SQL> @?/rdbms/admin/utlrp.sql
   ```

Relinking Executables

You can relink your product executables manually using the `relink` shell script located in the `$ORACLE_HOME/bin` directory. You must relink the product executables every time you apply an operating system patch or after an operating system upgrade. If you are running an Oracle Real Application Clusters installation, you must relink the product executables on each cluster node, unless the Oracle home directory is shared across the cluster using a cluster file system.

> **Note:** Shut down all executables that are running in the Oracle home directory that you are relinking. Also shut down applications linked with Oracle shared libraries.

Depending on the products that have been installed in the Oracle home directory, the `relink` script manually relinks Oracle product executables.

To relink product executables, enter the following command, where *`argument`* is one of the values listed in Table 3–1:

```
$ relink argument
```

Table 3–1 Relink Script Arguments

Argument	Description
`all`	Every product executable that has been installed
`oracle`	Oracle Database executable only
`network`	`net_client`, `net_server`, `cman`
`client`	`net_client`, `plsql`
`client_sharedlib`	Client shared library
`interMedia`	`ctx`
`ctx`	Oracle Text utilities
`precomp`	All precompilers that have been installed
`utilities`	All utilities that have been installed

Table 3–1 Relink Script Arguments (Cont.)

Argument	Description
`oemagent`	`oemagent` **Note:** To give the correct permissions to the `nmo` and `nmb` executables, you must run the `root.sh` script after relinking `oemagent`.
`ldap`	`ldap`, `oid`

4

Administering SQL*Plus

This chapter describes how to use and administer SQL*Plus on UNIX systems. It contains the following sections:

- Administering Command-Line SQL*Plus
- Using Command-Line SQL*Plus
- SQL*Plus Restrictions

> **See Also:** For more information about SQL*Plus, see the *SQL*Plus User's Guide and Reference*.

Administering Command-Line SQL*Plus

This section describes how to administer command-line SQL*Plus. In the examples in this section, SQL*Plus uses the value of the ORACLE_HOME environment variable wherever a question mark (?) appears.

Using Setup Files

When you start SQL*Plus, it executes the `glogin.sql` site profile set-up file and then executes the `login.sql` user profile set-up file.

Using the Site Profile File

The global site profile file is `$ORACLE_HOME/sqlplus/admin/glogin.sql`. If a site profile already exists at this location, it is overwritten when you install SQL*Plus. If SQL*Plus is removed, the site profile file is also removed.

Using the User Profile File

The user profile file is `login.sql`. SQL*Plus looks for this file in the current directory, and then in the directories specified by the SQLPATH environment variable. Set this environment variable to a colon-separated list of directories. SQL*Plus searches these directories for the `login.sql` file in the order that they are listed.

The options set in the `login.sql` file over-ride those set in the `glogin.sql` file.

> **See Also:** For more information about profile files, see the *SQL*Plus User's Guide and Reference*.

Using the PRODUCT_USER_PROFILE Table

Oracle Database provides the PRODUCT_USER_PROFILE table that you can use to disable the SQL and SQL*Plus commands that you specify. This table is created automatically when you choose an installation type that installs a preconfigured database.

> **See Also:** For more information about installation options, see the *Oracle Database Installation Guide for UNIX Systems*.

To recreate the PRODUCT_USER_PROFILE table, run the `$ORACLE_HOME/sqlplus/admin/pupbld.sql` script in the SYSTEM schema.

For example, enter the following commands, where *SYSTEM_PASSWORD* is the password of the SYSTEM user:

```
$ sqlplus SYSTEM/SYSTEM_PASSWORD
SQL> @?/sqlplus/admin/pupbld.sql
```

You can also recreate the PRODUCT_USER_PROFILE table manually in the SYSTEM schema by using the `$ORACLE_HOME/bin/pupbld` shell script. This script prompts for the SYSTEM password. If you need to run the `pupbld` script without interaction, set the SYSTEM_PASS environment variable to the SYSTEM user name and password.

Using Demonstration Tables

Oracle Database provides demonstration tables that you can use for testing. To install the demonstration tables in a database, you must choose an installation type that installs a preconfigured database.

> **See Also:** For more information about installation options, see the *Oracle Database Installation Guide for UNIX Systems*.

SQL*Plus Command-Line Help

This section describes how to install and remove the SQL*Plus command-line help.

> **See Also:** For more information about the SQL*Plus command-line help, see the *SQL*Plus User's Guide and Reference*.

Installing the SQL*Plus Command-Line Help

There are three ways to install the SQL*Plus command-line help:

- Complete an installation that installs a preconfigured database.

  When you install a preconfigured database as part of an installation, SQL*Plus automatically installs the SQL*Plus command-line help in the SYSTEM schema.

- Install the command-line help manually in the SYSTEM schema using the `$ORACLE_HOME/bin/helpins` shell script.

The `helpins` script prompts for the SYSTEM password. If you need to run this script without interaction, set the SYSTEM_PASS environment variable to the SYSTEM user name and password. For example:

- Bourne, Bash, or Korn shell:

```
$ SYSTEM_PASS=SYSTEM/system_password; export SYSTEM_PASS
```

- C or tcsh shell:

```
% setenv SYSTEM_PASS SYSTEM/system_password
```

- Install the command-line help manually in the SYSTEM schema using the `$ORACLE_HOME/sqlplus/admin/help/helpbld.sql` script.

  For example, enter the following commands, where *system_password* is the password of the SYSTEM user:

```
$ sqlplus SYSTEM/system_password
SQL> @?/sqlplus/admin/help/helpbld.sql ?/sqlplus/admin/help helpus.sql
```

> **Note:** Both the `helpins` shell script and the `helpbld.sql` script drop existing command-line help tables before creating new tables.

Removing the SQL*Plus Command-Line Help

To manually drop the SQL*Plus command-line help tables from the SYSTEM schema, run the `$ORACLE_HOME/sqlplus/admin/help/helpdrop.sql` script. For example, enter the following commands, where *system_password* is the password of the SYSTEM user:

```
$ sqlplus SYSTEM/system_password
SQL> @?/sqlplus/admin/help/helpdrop.sql
```

Using Command-Line SQL*Plus

This section describes how to use command-line SQL*Plus on UNIX systems.

Using a System Editor from SQL*Plus

If you enter an ED or EDIT command at the SQL*Plus prompt, the system starts an operating system editor, such as `ed`, `emacs`, `ned`, or `vi`. However, the PATH

environment variable must include the directory where the editor executable is located.

When you start the editor, the current SQL buffer is placed in the editor. When you exit the editor, the changed SQL buffer is returned to SQL*Plus.

You can specify which editor starts by defining the SQL*Plus _EDITOR variable. You can define this variable in the `glogin.sql` site profile, the `login.sql` user profile, or define it during the SQL*Plus session. For example, to set the default editor to `vi`, enter:

```
SQL> DEFINE _EDITOR=vi
```

If you do not set the _EDITOR variable, the value of either the EDITOR or the VISUAL environment variable is used. If both environment variables are set, the value of the EDITOR variable is used. When _EDITOR, EDITOR, and VISUAL are not specified, the default editor is `ed`.

If you start the editor, SQL*Plus uses the `afiedt.buf` temporary file to pass text to the editor. You can use the SET EDITFILE command to specify a different file name. For example, enter:

```
SQL> SET EDITFILE /tmp/myfile.sql
```

SQL*Plus does not delete the temporary file.

Running Operating System Commands from SQL*Plus

Using the HOST command or an exclamation mark (!) as the first character after the SQL*Plus prompt causes subsequent characters to be passed to a sub-shell. The SHELL environment variable sets the shell used to run operating system commands. The default shell is the Bourne shell (`/bin/sh`). If the shell cannot be run, SQL*Plus displays an error message.

To return to SQL*Plus, enter the `exit` command or press Ctrl+d.

For example, to run one command, enter:

```
SQL> ! command
```

In this example, *`command`* represents the operating system command that you want to run.

To run multiple operating system commands from SQL*Plus, enter the HOST or ! command then press Return. SQL*Plus returns you to the operating system prompt.

Interrupting SQL*Plus

While running SQL*Plus, you can stop the scrolling record display and terminate a SQL statement by pressing Ctrl+c.

Using the SPOOL Command

The default file extension of files generated by the SPOOL command is `.lst`. To change this extension, specify a spool file containing a period (.), for example:

```
SQL> SPOOL query.txt
```

SQL*Plus Restrictions

This section describes SQL*Plus restrictions.

Resizing Windows

The default values for the SQL*Plus LINESIZE and PAGESIZE system variables do not automatically adjust for window size.

Return Codes

UNIX return codes use only one byte, which is not enough space to return an Oracle error code. The range for a return code is 0 to 255.

Hiding Your Password

If you set the SYSTEM_PASS environment variable to the user name and password of the SYSTEM user, the output from the `ps` command might display this information. To prevent unauthorized access, enter the SYSTEM password only when prompted by SQL*Plus.

If you want to automatically run a script, consider using an authentication method that does not require you to store a password, for example, externally authenticated logins to Oracle Database. If you have a low security environment, you might consider using UNIX pipes in script files to pass a password to SQL*Plus, for example:

```
$ echo system_password | sqlplus SYSTEM @MYSCRIPT
```

Alternatively, enter the following lines at the command prompt:

```
$ sqlplus <<EOF
SYSTEM/system_password
SELECT ...
EXIT
EOF
```

In the preceding examples, *system_password* is the password of the SYSTEM user.

5

Configuring Oracle Net Services

This chapter describes how to configure Oracle Net Services on UNIX systems. It contains the following sections:

- Location of Oracle Net Services Configuration Files
- Adapters Utility
- Oracle Protocol Support
- Setting Up the Listener for TCP/IP or TCP/IP with SSL
- Oracle Advanced Security

> **See Also:** For more information about Oracle networking, see the *Oracle Net Services Administrator's Guide*.

Location of Oracle Net Services Configuration Files

Oracle Net Services configuration files are typically, but not always, located in the `$ORACLE_HOME/network/admin` directory. Depending on the file, Oracle Net uses a different search order to locate the file.

The search order for the `sqlnet.ora` and `ldap.ora` files is as follows:

1. The directory specified by the TNS_ADMIN environment variable, if set
2. The `$ORACLE_HOME/network/admin` directory

The search order for the `cman.ora`, `listener.ora`, and `tnsnames.ora` files is as follows:

1. The directory specified by the TNS_ADMIN environment variable, if set
2. One of the following directories:
   - For Solaris systems, the `/var/opt/oracle` directory
   - For other operating systems, the `/etc` directory
3. The `$ORACLE_HOME/network/admin` directory

For some system-level configuration files, users might have a corresponding user-level configuration file stored in their home directory. The settings in the user-level file override the settings in the system-level file. The following table lists the system-level configuration files and the corresponding user-level configuration files:

System-Level Configuration File	User-Level Configuration File
`sqlnet.ora`	`$HOME/.sqlnet.ora`
`tnsnames.ora`	`$HOME/.tnsnames.ora`

Sample Configuration Files

The `$ORACLE_HOME/network/admin/samples` directory contains samples of the `cman.ora`, `listener.ora`, `sqlnet.ora`, and `tnsnames.ora` configuration files.

> **Note:** The `cman.ora` file is installed only if you selected Connection Manager as part of a custom installation.

Adapters Utility

Use the `adapters` utility to display the transport protocols, naming methods, and Oracle Advanced Security options that Oracle Database supports on your system. To use the `adapters` utility, enter the following commands:

```
$ cd $ORACLE_HOME/bin
$ adapters ./oracle
```

The `adapters` utility displays output similar to the following:

```
Oracle Net transport protocols linked with ./oracle are

         IPC
         BEQ
         TCP/IP
         SSL
         RAW

Oracle Net naming methods linked with ./oracle are:

         Local Naming (tnsnames.ora)
         Oracle Directory Naming
         Oracle Host Naming
         NIS Naming


Oracle Advanced Security options linked with ./oracle are:

         RC4 40-bit encryption
         RC4 128-bit encryption
         RC4 256-bit encryption
         DES40 40-bit encryption
         DES 56-bit encryption
         3DES 112-bit encryption
         3DES 168-bit encryption
         AES 128-bit encryption
         AES 192-bit encryption
         SHA crypto-checksumming (for FIPS)
         SHA-1 crypto-checksumming
         Kerberos v5 authentication
         RADIUS authentication
         ENTRUST authentication
```

On an Oracle client system, the `adapters` utility displays the configured Oracle transport protocols, naming methods, and security options. To run the `adapters` utility on an Oracle client system:

```
$ cd $ORACLE_HOME/bin
$ adapters
```

The `adapters` utility displays output similar to the following:

```
Installed Oracle Net transport protocols are:

         IPC
         BEQ
         TCP/IP
         SSL
         RAW

Installed Oracle Net naming methods are:

         Local Naming (tnsnames.ora)
         Oracle Directory Naming
         Oracle Host Naming
         NIS Naming


Installed Oracle Advanced Security options are:

         RC4 40-bit encryption
         RC4 56-bit encryption
         RC4 128-bit encryption
         RC4 256-bit encryption
         DES40 40-bit encryption
         DES 56-bit encryption
         3DES 112-bit encryption
         3DES 168-bit encryption
         AES 128-bit encryption
         AES 192-bit encryption
         AES 256-bit encryption
         MD5 crypto-checksumming
         SHA-1 crypto-checksumming
         Kerberos v5 authentication
         RADIUS authentication
```

See Also: For more information about the `adapters` utility, see the *Oracle Net Services Administrator's Guide*.

Oracle Protocol Support

Oracle protocol support is a component of Oracle Net. It includes the following:

- IPC protocol support
- TCP/IP protocol support
- TCP/IP with SSL protocol support

The IPC, TCP/IP, and TCP/IP with SSL protocol supports each have an address specification that is used in Oracle Net Services configuration files and in the DISPATCHER initialization parameter. The following sections describe the address specifications for each of the protocol supports.

Note: On HP-UX (PA-RISC) and Tru64 UNIX systems, you can use DCE as an Oracle Net protocol, if it is installed. For more information about configuring the DCE protocol support, see the *Oracle Advanced Security Administrator's Guide*.

See Also: For more information about Oracle protocol support, see the *Oracle Net Services Administrator's Guide*.

IPC Protocol Support

The IPC protocol support can be used only when the client program and Oracle Database are installed on the same system. This protocol support requires a listener. It is installed and linked to all client tools and the `oracle` executable.

The IPC protocol support requires an address specification in the following format:

```
(ADDRESS = (PROTOCOL=IPC)(KEY=key))
```

The following table describes the parameters used in this address specification:

Parameter	Description
PROTOCOL	The protocol to be used. The value is IPC. It is not case sensitive.
KEY	Any name unique from any other name used for an IPC KEY on the same system.

The following example shows a sample IPC protocol address:

```
(ADDRESS= (PROTOCOL=IPC)(KEY=EXTPROC))
```

TCP/IP Protocol Support

TCP/IP is the standard communication protocol used for client/server communication over a network. The TCP/IP protocol support enables communication between client programs and the Oracle Database, whether they are installed on the same or different systems. If the TCP/IP protocol is installed on your system, the TCP/IP protocol support is installed and linked to all client tools and to the `oracle` executable.

The TCP/IP protocol support requires an address specification in the following format:

```
(ADDRESS = (PROTOCOL=TCP)(HOST=hostname)(PORT=port))
```

The following table describes the parameters used in this address specification:

Parameter	Description
PROTOCOL	The protocol support to be used. The value is TCP. It is not case sensitive.
HOST	The host name or the host IP address.
PORT	The TCP/IP port. Specify the port as either a number or the alias name mapped to the port in the `/etc/services` file. Oracle recommends a value of 1521.

The following example shows a sample TCP/IP protocol address:

```
(ADDRESS= (PROTOCOL=TCP)(HOST=MADRID)(PORT=1521))
```

TCP/IP with SSL Protocol Support

The TCP/IP with SSL protocol support enables an Oracle application on a client to communicate with remote Oracle databases through TCP/IP and SSL. To use TCP/IP with SSL, Oracle Advanced Security must be installed.

The TCP/IP with SSL protocol support requires an address specification in the following format:

```
(ADDRESS = (PROTOCOL=TCPS)(HOST=hostname)(PORT=port))
```

The following table describes the parameters used in this address specification:

Parameter	Description
PROTOCOL	The protocol to be used. The value is TCPS. It is not case sensitive.
HOST	The host name or the host IP address.
PORT	The TCP/IP with SSL port. Specify the port as either a number or the alias name mapped to the port in the `/etc/services` file. Oracle recommends a value of 2484.

The following example shows a sample TCP/IP with SSL protocol address:

```
(ADDRESS= (PROTOCOL=TCPS)(HOST=MADRID)(PORT=2484))
```

Setting Up the Listener for TCP/IP or TCP/IP with SSL

Oracle recommends that you reserve a port for the listener in the `/etc/services` file of each Oracle Net Services node on the network. The default port is 1521. The entry lists the listener name and the port number, for example:

```
oraclelistener    1521/tcp
```

In this example, *`oraclelistener`* is the name of the listener as defined in the `listener.ora` file. Reserve more than one port if you intend to start more than one listener.

If you intend to use SSL, you should define a port for TCP/IP with SSL in the `/etc/services` file. Oracle recommends a value of 2484. For example:

```
oraclelistenerssl     2484/tcps
```

In this example, *`oraclelistenerssl`* is the name of the listener as defined in the `listener.ora` file. Reserve more than one port if you intend to start more than one listener.

Oracle Advanced Security

When you install Oracle Advanced Security, three `.bak` files are created: `naeet.o.bak`, `naect.o.bak`, and `naedhs.o.bak`. These files are located in the `$ORACLE_HOME/lib` directory. They are required for relinking if you decide to remove Oracle Advanced Security. Do not delete them.

6

Using Oracle Precompilers and the Oracle Call Interface

This chapter describes Oracle Precompilers and the Oracle Call Interface. It contains the following sections:

> **Note:** To use the demonstrations described in this chapter, you must install the Oracle Database Examples included on the Oracle Database 10*g* Companion CD.

See Also: For information about using SQL*Plus to create demonstration tables, see the "Using Demonstration Tables" section on page 4-3.

Overview of Oracle Precompilers

Oracle precompilers are application-development tools used to combine SQL statements for an Oracle database with programs written in a high-level language. Oracle precompilers are compatible with ANSI SQL and are used to develop open, customized applications that run with Oracle Database or any other ANSI SQL database management system.

Precompiler Configuration Files

Configuration files for the Oracle precompilers are located in the `$ORACLE_HOME/precomp/admin` directory. Table 6–1 lists the names of the configuration files for each precompiler.

Table 6–1 System Configuration Files for Oracle Precompilers

Product	Configuration File
Pro*C/C++ release 10.1.0.2.0	`pcscfg.cfg`
Pro*COBOL release 10.1.0.2.0 (AIX, HP-UX, Linux x86, Solaris, and Tru64 UNIX only)	`pcbcfg.cfg`
Pro*FORTRAN release 1.8.78 (AIX, HP-UX, Solaris, Tru64 UNIX)	`pccfor.cfg`
Object Type Translator release 10.1.0.2.0	`ottcfg.cfg`
Oracle SQL*Module for Ada release 10.1.0.2.0 (AIX only)	`pmscfg.cfg`

Relinking Precompiler Executables

Use the `$ORACLE_HOME/precomp/lib/ins_precomp.mk` make file to relink all precompiler executables. To manually relink a particular precompiler executable, enter the following command:

```
$ make -f ins_precomp.mk executable
```

This command creates the new executable in the `$ORACLE_HOME/precomp/lib` directory, and then moves it to the `$ORACLE_HOME/bin` directory.

In the preceding example, *`executable`* is a product executable listed in Table 6–2.

Table 6–2 Products and Their Corresponding Executable

Product	Executable
Pro*C/C++ release 10.1.0.2.0	`proc`
Pro*COBOL release 10.1.0.2.0 (AIX, HP-UX, Linux x86, Solaris, and Tru64 UNIX)	`procob` or `rtsora`
Pro*COBOL release 10.1.0.2.0 32-bit (AIX, HP-UX PA-RISC, and Solaris only)	`procob32` or `rtsora32`
Pro*FORTRAN release 1.8.78 (AIX, HP-UX, Solaris, and Tru64 UNIX only)	`profor`
Pro*FORTRAN release 1.8.78 32-bit (HP-UX only)	`profor32`
Oracle SQL*Module for Ada release 10.1.0.2.0 (AIX only)	`modada`

Precompiler README Files

Table 6–3 lists the location of the precompiler README files. The README files describe changes made to the precompiler since the last release.

Table 6–3 Location of Precompiler README Files

Precompiler	README File
Pro*C/C++ release 10.1.0.2.0	`$ORACLE_HOME/precomp/doc/proc2/readme.doc`
Pro*COBOL release 10.1.0.2.0	`$ORACLE_HOME/precomp/doc/procob2/readme.doc`
Pro*FORTRAN release 1.8.78.0.0	`$ORACLE_HOME/precomp/doc/pro1x/readme.txt`

Issues Common to All Precompilers

The following issues are common to all precompilers.

Uppercase to Lowercase Conversion

In languages other than C, the compiler converts an uppercase function or subprogram name to lowercase. This can cause a `No such user exists` error message. If you receive this error message, verify that the function or subprogram name in your option file matches the case used in the IAPXTB table.

Vendor Debugger Programs

Precompilers and vendor-supplied debuggers can be incompatible. Oracle does not guarantee that a program run using a debugger will perform the same way when it is run without the debugger.

Value of IRECLEN and ORECLEN

The IRECLEN and ORECLEN parameters do not have maximum values.

Static and Dynamic Linking

You can statically or dynamically link Oracle libraries with precompiler and OCI or OCCI applications. With static linking, the libraries and objects of the whole application are linked together into a single executable program. As a result, application executables can become very large.

With dynamic linking, the executing code is partly stored in the executable program and partly stored in libraries that are linked dynamically by the application at runtime. Libraries that are linked at runtime are called dynamic or shared libraries. The benefits of dynamic linking are:

- Smaller disk space requirements—More than one application or invocation of the same application can use the same dynamic library.
- Smaller main memory requirements—The same dynamic library image is loaded into main memory only once and it can be shared by more than one application.

Client Shared and Static Libraries

The client shared and static libraries are located in the `$ORACLE_HOME/lib` directory. If you use the Oracle provided `demo_`*`product`*`.mk` make file to link an application, the client shared library is linked by default.

You might see an error message similar to the following when starting an executable:

- On AIX systems:

```
$ sample1
exec(): 0509-036 Cannot load program ./sample1 because of the following
errors:
0509-022 Cannot load library libclntsh.a [shr.o]
0509-026 System error: A file or directory in the pathname does not exist.
```

- On HP-UX PA-RISC systems:

```
$ sample1
/usr/lib/dld.sl: Can't open shared library:
/u01/app/oracle/product/10.1.0/lib/libclntsh.sl.10.1
/usr/lib/dld.sl: No such file or directory
Abort (core dumped)
```

- On Solaris and Linux systems:

```
$ sample1
ld.so.1: sample1: fatal: libclntsh.so.10.1: can't open file: errno=2
Killed
```

- On Tru64 UNIX systems:

```
$ sample1
/sbin/loader: Fatal Error: Cannot map libclntsh.so
Killed
```

If you see a similar error message, set one of the following environment variables, depending on your platform:

Platform	Environment Variable	Sample Setting
AIX (64-bit applications)	LIBPATH	`$ORACLE_HOME/lib`
AIX (32-bit applications)	LIBPATH	`$ORACLE_HOME/lib32`
HP-UX (64-bit applications), Linux, Solaris, and Tru64 UNIX	LD_LIBRARY_PATH	`$ORACLE_HOME/lib`
HP-UX (32-bit applications)	SHLIB_PATH	`$ORACLE_HOME/lib32`

The client shared library is created automatically during installation. If you need to recreate it, follow these steps:

1. Exit all client applications that use the client shared library, including all Oracle client applications such as SQL*Plus and Recovery Manager.
2. Log in as the `oracle` user and enter the following command:

```
$ $ORACLE_HOME/bin/genclntsh
```

HP-UX PA-RISC Only: Non-threaded Client Shared Library

On HP-UX PA-RISC, you can use a non-threaded client shared library. However, you cannot use this library with any OCI application that uses or has a dependency on threads.

To use this library for applications that do not use threads, use one of the following commands to build your OCI application:

- For 32-bit applications:

```
$ make -f demo_rdbms32.mk build_nopthread EXE=oci02 OBJS=oci02.o
```

- For 64-bit applications:

```
$ make -f demo_rdbms.mk build_nopthread EXE=oci02 OBJS=oci02.o
```

Support for 32-Bit and 64-Bit Client Applications

The following table identifies the types of client application (32-bit, 64-bit, or both) supported on each platform:

Client Application Type	Supported Platforms
32-bit only	Linux x86
64-bit only	Linux Itanium and Tru64 UNIX
32-bit and 64-bit	AIX, HP-UX, Solaris

By default on platforms that support both 32-bit and 64-bit client applications, all demonstrations and client applications provided with Oracle Database 10*g* release 1 (10.1) link and run in 64-bit mode. However, on these platforms, you can build 32-bit and 64-bit client applications in the same Oracle home directory. The following table lists the 32-bit and 64-bit client shared libraries for these platforms:

Platform	32-Bit Client Shared Library	64-Bit Client Shared Library
AIX	`$ORACLE_HOME/lib32/libclntsh.a`	`$ORACLE_HOME/lib/libclntsh.a`
	`$ORACLE_HOME/lib32/libclntsh.so`	`$ORACLE_HOME/lib/libclntsh.so`
HP-UX PA-RISC	`$ORACLE_HOME/lib32/libclntsh.sl`	`$ORACLE_HOME/lib/libclntsh.sl`
Solaris and HP-UX Itanium	`$ORACLE_HOME/lib32/libclntsh.so`	`$ORACLE_HOME/lib/libclntsh.so`

To implement a mixed word-size installation:

1. To generate the 32-bit and 64-bit client shared libraries, enter the following command:

```
$  $ORACLE_HOME/bin/genclntsh
```

2. Include the paths of the required 32-bit and 64-bit client shared libraries in one of the following environment variables, depending on your platform:

Platform	Environment Variable
AIX	LIBPATH
HP-UX (32-bit client applications)	SHLIB_PATH
HP-UX and Solaris (64-bit client applications)	LD_LIBRARY_PATH

Building 32-Bit Pro*C and OCI Customer Applications

On platforms that support both 32-bit and 64-bit Pro*C and Oracle Call Interface (OCI) customer applications, you can find more information about building 32-bit Pro*C and OCI applications in the following files:

For information about...	See the Following Make Files:
Building 32-bit Pro*C applications	`$ORACLE_HOME/precomp/demo/proc/demo_proc32.mk`
Building 32-bit OCI applications	`$ORACLE_HOME/rdbms/demo/demo_rdbms32.mk`

AIX, HP-UX, and Solaris Only: 32-Bit Executables and Directories

On platforms that support both 32-bit and 64-bit applications, the following directories contain 32-bit executables and libraries:

- `$ORACLE_HOME/lib32`
- `$ORACLE_HOME/rdbms/lib32`
- `$ORACLE_HOME/hs/lib32`
- `$ORACLE_HOME/network/lib32`

- $ORACLE_HOME/precomp/lib32
- $ORACLE_HOME/bin

Pro*C/C++ Precompiler

Before you use the Pro*C/C++ precompiler, verify that the correct version of the operating system compiler is properly installed.

> **See Also:** For information about the required compiler versions on each platform, see the *Oracle Database Installation Guide for UNIX Systems* and for more information about the Pro*C/C++ precompiler and interface features, see the *Pro*C/C++ Precompiler Programmer's Guide*.

Pro*C/C++ Demonstration Programs

Demonstration programs are provided to show the features of the Pro*C/C++ precompiler. There are three types of demonstration programs: C, C++, and Object programs. All of the demonstration programs are located in the $ORACLE_HOME/precomp/demo/proc directory. By default, all programs are dynamically linked with the client shared library.

To run, the programs require the demonstration tables created by the $ORACLE_HOME/sqlplus/demo/demobld.sql script to exist in the SCOTT schema with the password TIGER.

> **Note:** You must unlock the SCOTT account and set the password before creating the demonstrations.

Use the demo_proc.mk make file, located in the $ORACLE_HOME/precomp/demo/proc/ directory, to create the demonstration programs. For example, to precompile, compile, and link the sample1 demonstration program, enter the following command:

```
$ make -f demo_proc.mk sample1
```

To create all of the C demonstration programs for Pro*C/C++, enter:

```
$ make -f demo_proc.mk samples
```

To create all of the C++ demonstration programs for Pro*C/C++, enter:

```
$ make -f demo_proc.mk cppsamples
```

To create all of the Object demonstration programs for Pro*C/C++, enter:

```
$ make -f demo_proc.mk object_samples
```

Some demonstration programs require you to run a SQL script, located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, a message appears requesting you to run it. To build a demonstration program and run the corresponding SQL script, include the `make` macro argument `RUNSQL=run` on the command line. For example, to create the `sample9` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/sample9.sql` script, enter:

```
$ make -f demo_proc.mk sample9 RUNSQL=run
```

To create all of the Object demonstration programs and run all corresponding required SQL scripts, enter:

```
$ make -f demo_proc.mk object_samples RUNSQL=run
```

Pro*C/C++ User Programs

Note: On all platforms except Linux x86, the `demo_proc.mk` make file builds 64-bit user programs by default. On all platforms except Linux Itanium and Tru64 UNIX, you can also use the `demo_proc32.mk` make file to build 32-bit user programs. For more information about creating 32-bit user programs, see the make file.

You can use the `$ORACLE_HOME/precomp/demo/proc/demo_proc.mk` make file to create user programs. To create a program using the `demo_proc.mk` make file, enter a command similar to the following:

```
$ make -f demo_proc.mk target OBJS="objfile1 objfile2 ..." EXE=exename
```

In this example:

- *target* is the make file target that you want to use
- *objfilen* is the object file to link the program
- *exename* is the executable program

For example, to create the program `myprog` from the Pro*C/C++ source file `myprog.pc`, enter one of the following commands, depending on the source and the type of executable that you want to create:

- For C source, dynamically linked with the client shared library, enter:

```
$ make -f demo_proc.mk build OBJS=myprog.o EXE=myprog
```

- For C source, statically linked with the client shared library, enter:

```
$ make -f demo_proc.mk build_static OBJS=myprog.o EXE=myprog
```

- For C++ source, dynamically linked with the client shared library, enter:

```
$ make -f demo_proc.mk cppbuild OBJS=myprog.o EXE=myprog
```

- For C++ source, statically linked with the client shared library, enter:

```
$ make -f demo_proc.mk cppbuild_static OBJS=myprog.o EXE=myprog
```

Pro*COBOL Precompiler

Note: Pro*COBOL is not supported on Linux Itanium.

Table 6–4 shows the naming conventions for the Pro*COBOL precompiler.

Table 6–4 Pro*COBOL Naming Conventions

Item	Pro*COBOL 10.1.0.2.0
Executable	`procob` or `procob32`
Demonstration Directory	`procob2`
Make file for Micro Focus Server Express COBOL	`demo_procob.mk`

Pro*COBOL supports statically linked, dynamically linked, or dynamically loadable programs. Dynamically linked programs use the client shared library. Dynamically loadable programs use the `rtsora` executable (or the `rtsora32` executable for 32-bit COBOL compilers) located in the `$ORACLE_HOME/bin` directory.

Pro*COBOL Environment Variables

This section describes the environment variables required by Pro*COBOL.

If the LD_LIBRARY_PATH, LIBPATH, or SHLIB_PATH environment variable setting does not include the `$COBDIR/coblib` directory, an error message similar to the following appears when you compile a program:

- On Linux x86:

```
rtsora: error while loading shared libraries: libcobrts_t.so:
cannot open shared object file: No such file or directory
```

- On Tru64 UNIX:

```
356835:rtsora: /sbin/loader: Fatal Error:
Cannot map library libcobrts64_t.so.2
```

- On AIX, HP-UX PA-RISC, and Solaris:

```
ld.so.1: rts32: fatal: libfhutil.so.2.0: Can't open file: errno=2
```

- On HP-UX Itanium:

```
/usr/lib/hpux64/dld.so: Unable to find library 'libcobrts64_t.so.2'.
Killed
```

Micro Focus Server Express COBOL Compiler

For the Micro Focus Server Express COBOL compiler, you must set the COBDIR environment variable and the LD_LIBRARY_PATH, LIBPATH, or SHLIB_PATH environment variable, depending on your operating system.

COBDIR Set the COBDIR environment variable to the directory where the compiler is installed. For example, if the compiler is installed in the `/opt/cobol` directory, enter:

- For the Bourne, Bash, or Korn shell:

```
$ COBDIR=${COBDIR}:/opt/cobol
$ export COBDIR
```

- For the C or tcsh shell:

```
% setenv COBDIR ${COBDIR}:/opt/cobol
```

Solaris and HP-UX with 64-Bit Applications Only: LD_LIBRARY_PATH Set the LD_LIBRARY_PATH environment variable to the directory where the compiler library is installed. For example, if the compiler library is installed in the `$COBDIR/coblib` directory, enter:

- For the Bourne, Bash, or Korn shell:

```
$ LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$COBDIR/coblib
$ export LD_LIBRARY_PATH
```

- For the C or tcsh shell:

```
% setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:$COBDIR/coblib
```

AIX Only: LIBPATH Set the LIBPATH environment variable to the directory where the compiler library is installed. For example, if the compiler library is installed in the `$COBDIR/coblib` directory, enter:

- For the Bourne, Bash, or Korn shell:

```
$ LIBPATH=${LIBPATH}:$COBDIR/coblib
$ export LIBPATH
```

- For the C or tcsh shell:

```
% setenv LIBPATH ${LIBPATH}:$COBDIR/coblib
```

HP-UX PA-RISC (32-Bit Applications): SHLIB_PATH Set the SHLIB_PATH environment variable to the directory where the compiler library is installed. For example, if the compiler library is installed in the `$COBDIR/coblib` directory, enter:

- For the Bourne, Bash, or Korn shell:

```
$ SHLIB_PATH=${SHLIB_PATH}:$COBDIR/coblib
$ export SHLIB_PATH
```

- For the C or tcsh shell:

```
% setenv SHLIB_PATH ${SHLIB_PATH}:$COBDIR/coblib
```

Pro*COBOL Oracle Runtime System

Oracle provides its own complete runtime system, called `rtsora` (or `rtsora32` for 32-bit COBOL compilers on 64-bit systems) to run dynamically loadable Pro*COBOL programs. Use the `rtsora` (or `rtsora32`) runtime system instead of the `cobrun` runtime system to run dynamically loadable Pro*COBOL programs. If you attempt to run a Pro*COBOL program with `cobrun`, you see an error message similar to the following:

```
$ cobrun sample1.gnt
Load error : file 'SQLADR'
error code: 173, pc=0, call=1, seg=0
173     Called program file not found in drive/directory
```

Pro*COBOL Demonstration Programs

Demonstration programs are provided to show the features of the Pro*COBOL precompiler. The demonstration programs are located in the `$ORACLE_HOME/precomp/demo/procob2` directory. By default, all programs are dynamically linked with the client shared library.

To run the programs, the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script must exist in the SCOTT schema with the password TIGER.

> **Note:** You must unlock the SCOTT account and set the password before creating the demonstrations.

Use the following make file to create the demonstration programs:

```
$ORACLE_HOME/precomp/demo/procob2/demo_procob.mk
```

To precompile, compile, and link the `sample1` demonstration program for Pro*COBOL, enter:

```
$ make -f demo_procob.mk sample1
```

To create all of the Pro*COBOL demonstration programs, enter:

```
$ make -f demo_procob.mk samples
```

To create and run a dynamically loadable `sample1.gnt` program to be used with the `rtsora` runtime system, enter:

```
$ make -f demo_procob.mk sample1.gnt
$ rtsora sample1.gnt
```

Some demonstration programs require you to run a SQL script, located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, a message appears requesting you to run it. To build a demonstration program and run the corresponding SQL script, include the `make` macro argument `RUNSQL=run` on the command line. For example, to create the `sample9` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/sample9.sql` script, enter:

```
$ make -f demo_procob.mk sample9 RUNSQL=run
```

To create all of the Pro*COBOL demonstration programs and run all required SQL scripts, enter:

```
$ make -f demo_procob.mk samples RUNSQL=run
```

Pro*COBOL User Programs

Note: On all platforms except Linux x86, the `demo_procob.mk` make file builds 64-bit user programs by default. On all platforms except HP-UX Itanium and Tru64 UNIX, you can also use the `demo_procob_32.mk` make file to build 32-bit user programs. For more information about creating 32-bit user programs, see the make file.

You can use the demonstration make file to create user programs. To create a program using the demonstration make file, enter a command similar to the following:

```
$ make -f demo_procob.mk target COBS="cobfile1 cobfile2 ..." EXE=exename
```

In this example:

- *target* is the make file target that you want to use
- *cobfilen* is the COBOL source file for the program
- *exename* is the executable program

For example, to create the program myprog, enter one of the following commands, depending on the source and type of executable that you want to create:

- For COBOL source, dynamically linked with the client shared library, enter:

```
$ make -f demo_procob.mk build COBS=myprog.cob EXE=myprog
```

- For COBOL source, statically linked, enter:

```
$ make -f demo_procob.mk build_static COBS=myprog.cob EXE=myprog
```

- For COBOL source, dynamically loadable for use with rtsora, (or rtsora32 for 32-bit COBOL compilers) enter:

```
$ make -f demo_procob.mk myprog.gnt
```

FORMAT Precompiler Option

The FORMAT precompiler option specifies the format of input lines for COBOL. If you specify the default value ANSI, columns 1 to 6 contain an optional sequence number, column 7 indicates comments or continuation lines, paragraph names begin in columns 8 to 11, and statements begin in columns 12 to 72.

If you specify the value TERMINAL, columns 1 to 6 are dropped, making column 7 the left-most column.

Pro*FORTRAN Precompiler

Note: Pro*FORTRAN is not supported on Linux.

Before you use the Pro*FORTRAN precompiler, verify that the correct version of the compiler is installed.

See Also: For information about the required compiler versions for each platform, see the *Oracle Database Installation Guide for UNIX Systems*, and for more information about Pro*FORTRAN precompiler and interface features, see the *Pro*FORTRAN Precompiler Programmer's Guide*.

Pro*FORTRAN Demonstration Programs

Demonstration programs are provided to show the features of the Pro*FORTRAN precompiler. All of the demonstration programs are located in the `$ORACLE_HOME/precomp/demo/profor` directory. By default, all programs are dynamically linked with the client shared library.

To run the programs, the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script must exist in the SCOTT schema with the password TIGER.

Note: You must unlock the SCOTT account and set the password before creating the demonstrations.

Use the `demo_profor.mk` make file, located in the `$ORACLE_HOME/precomp/demo/profor` directory, to create the demonstration programs. For example, to precompile, compile, and link the `sample1` demonstration program, enter:

```
$ make -f demo_profor.mk sample1
```

To create all of the Pro*FORTRAN demonstration programs, enter:

```
$ make -f demo_profor.mk samples
```

Some demonstration programs require you to run a SQL script, located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, a message appears requesting you to run it. To build a demonstration program and run the corresponding SQL script, include the `make` macro argument `RUNSQL=run` on the command line. For example, to create the `sample11` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/sample11.sql` script, enter:

```
$ make -f demo_profor.mk sample11 RUNSQL=run
```

To create all of the Pro*FORTRAN demonstration programs and run all required SQL scripts, enter:

```
$ make -f demo_profor.mk samples RUNSQL=run
```

Pro*FORTRAN User Programs

> **Note:** On HP-UX, the `demo_profor.mk` make file builds 64-bit user programs by default. On other platforms, it builds 32-bit user programs. You can also use the `demo_profor_32.mk` make file on HP-UX to build 32-bit user programs. For more information about creating 32-bit user programs, see the make file.

You can use the `$ORACLE_HOME/precomp/demo/profor/demo_profor.mk` make file to create user programs. To create a program using the `demo_proc.mk` make file, enter a command similar to the following:

```
$ make -f demo_profor.mk target FORS="forfile1 forfile2 ..." EXE=exename
```

In this example:

- *`target`* is the make file target that you want to use
- *`forfilen`* is the FORTRAN source for the program
- *`exename`* is the executable program

For example, to create the program `myprog` from the Pro*FORTRAN source file `myprog.pfo`, enter one of the following commands, depending on the type of executable that you want to create:

- For an executable dynamically linked with the client shared library, enter:

  ```
  $ make -f demo_profor.mk build FORS=myprog.f EXE=myprog
  ```

- For an executable statically linked, enter:

  ```
  $ make -f demo_profor.mk build_static FORS=myprog.f EXE=myprog
  ```

AIX Only: SQL*Module for Ada

Before using SQL*Module for Ada, verify that the correct version of the compiler is installed.

> **See Also:** For more information about the required compiler versions on each platform, see the *Oracle Database Installation Guide for UNIX Systems*, and for more information about SQL*Module for Ada, see the *SQL*Module for Ada Programmer's Guide*.

SQL*Module for Ada Demonstration Programs

Demonstration programs are provided to show the features of SQL*Module for Ada. All of the demonstration programs are located in the `$ORACLE_HOME/precomp/demo/modada` directory. By default, all programs are dynamically linked with the client shared library.

To run the `ch1_drv` demonstration program the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script must exist in the SCOTT schema with the password TIGER.

> **Note:** You must unlock the SCOTT account and set the password before creating the demonstrations.

The `demcalsp` and `demohost` demonstration programs require that the sample college database exists in the MODTEST schema. You can use the appropriate `make` command to create the MODTEST schema and load the sample college database.

To create all of the SQL*Module for Ada demonstration programs, run the necessary SQL scripts to create the MODTEST user, and create the sample college database, enter:

```
$ make -f demo_modada.mk all RUNSQL=run
```

To create a single demonstration program (`demohost`), and run the necessary SQL scripts to create the MODTEST user, and create the sample college database, enter:

```
$ make -f demo_modada.mk makeuser loaddb demohost RUNSQL=run
```

To create all of the SQL*Module for Ada demonstration programs, without recreating the sample college database, enter:

```
$ make -f demo_modada.mk samples
```

To create a single demonstration program (`demohost`), without recreating the sample college database, enter:

```
$ make -f demo_modada.mk demohost
```

All programs require that an Oracle Net connect string or alias named INST1_ALIAS is defined and is capable of connecting to the database where the appropriate tables exist.

SQL*Module for Ada User Programs

You can use the `$ORACLE_HOME/precomp/demo/modada/demo_modada.mk` make file to create user programs. To create a user program with the `demo_modada.mk` make file, enter a command similar to the following:

```
$ make -f demo_modada.mk ada OBJS="module1 module2 ..." \
EXE=exename MODARGS=SQL*Module_arguments
```

In this example:

- *`modulen`* is a compiled Ada object
- *`exename`* is the executable program
- *`SQL*Module_arguments`* are the command-line arguments to be passed to the SQL*Module

> **See Also:** For more information about SQL*Module for Ada user programs, see the *SQL*Module for Ada Programmers Guide*.

Oracle Call Interface and Oracle C++ Call Interface

Before you use the Oracle Call Interface (OCI) or Oracle C++ Call Interface (OCCI), verify that the correct version of C or C++ is installed.

> **See Also:** For more information about the required version of C and C++ for your operating system, see the *Oracle Database Installation Guide for UNIX Systems*. For more information about the OCI and OCCI, see the *Oracle Call Interface Programmer's Guide* or the *Oracle C++ Call Interface Programmer's Guide*.

OCI and OCCI Demonstration Programs

Demonstration programs that show the features of OCI and OCCI are provided. There are two types of demonstration programs: C and C++. All of the demonstration programs are located in the `$ORACLE_HOME/rdbms/demo` directory. By default, all programs are dynamically linked with the client shared library.

To run the demonstration programs, the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script must exist in the SCOTT schema with the password TIGER.

> **Note:** You must unlock the SCOTT account and set the password before creating the demonstrations.

Use the `demo_rdbms.mk` make file, located in the `$ORACLE_HOME/rdbms/demo` directory, to create the demonstration programs. For example, to compile and link the `cdemo1` demonstration program, enter the following command:

```
$ make -f demo_rdbms.mk cdemo1
```

To create all of the C demonstration programs for OCI, enter:

```
$ make -f demo_rdbms.mk demos
```

To create all of the C++ demonstration programs for OCCI, enter:

```
$ make -f demo_rdbms.mk c++demos
```

OCI and OCCI User Programs

> **Note:** On all platforms except Linux x86, the `demo_rdbms.mk` make file builds 64-bit user programs by default. On all platforms except Linux Itanium and Tru64 UNIX, you can use the `demo_rdbms32.mk` make file to build 32-bit user programs. For more information about creating 32-bit user programs, see the make file.

You can use the `$ORACLE_HOME/rdbms/demo/demo_rdbms.mk` make file to create user programs. The syntax for creating a user program with the `demo_rdbms.mk` make file is:

```
$ make -f demo_rdbms.mk target OBJS="objfile1 objfile2 ..." EXE=exename
```

In the preceding example:

- *`target`* is the make file target that you want to use
- *`objfilen`* is the object file to link the program
- *`exename`* is the executable program

For example, to create the `myprog` program from the C source `myprog.c`, enter one of the following commands, depending on the type of executable that you want to create:

- For C source, dynamically linked with the client shared library, enter:

  ```
  $ make -f demo_rdbms.mk build OBJS=myprog.o EXE=myprog
  ```

- For C source, statically linked, enter:

  ```
  $ make -f demo_rdbms.mk build_static OBJS=myprog.o EXE=myprog
  ```

For example, to create the `myprog` program from the C++ source `myprog.cpp`, enter one of the following commands, depending on the type of executable that you want to create:

- For C++ source, dynamically linked with the client shared library, enter:

  ```
  $ make -f demo_rdbms.mk buildc++ OBJS=myprog.o EXE=myprog
  ```

- For C++ source, statically linked, enter:

  ```
  $ make -f demo_rdbms.mk buildc++_static OBJS=myprog.o EXE=myprog
  ```

Oracle JDBC/OCI Programs With a 64-Bit Driver

> **Note:** You can also use the instructions and make files described in this section to create your own JDBC/OCI user programs that use a 64-bit driver.

To run JDBC/OCI demonstration programs with a 64-bit driver, follow these steps:

1. Add `$ORACLE_HOME/jdbc/lib/ojdbc14.jar` to the start of the CLASSPATH environment variable for each of the following files:

   ```
   jdbc/demo/samples/jdbcoci/Makefile
   jdbc/demo/samples/generic/Inheritance/Inheritance1/Makefile
   jdbc/demo/samples/generic/Inheritance/Inheritance2/Makefile
   jdbc/demo/samples/generic/Inheritance/Inheritance3/Makefile
   jdbc/demo/samples/generic/JavaObject1/Makefile
   jdbc/demo/samples/generic/NestedCollection/Makefile
   ```

2. In the `$ORACLE_HOME/jdbc/demo/samples/generic/Makefile` file, modify the JAVA and JAVAC variables to specify the JDK location as follows, depending on your platform:

   - For Solaris and HP-UX, change JAVA and JAVAC to the following, specifying the `-d64` flag:

     ```
     JAVA=${ORACLE_HOME}/java/bin/java -d64
     JAVAC=${ORACLE_HOME}/java/bin/javac -d64
     ```

   - For AIX, change JAVA and JAVAC to the following, where *`JDK14_HOME`* is the directory in which the 64-bit JDK is installed:

     ```
     JAVA=$(JDK14_HOME)/bin/java
     JAVAC=$(JDK14_HOME)/bin/javac
     ```

3. In the `jdbc/demo/samples/generic/Makefile` file, replace all occurrences of *`JDK14_HOME`*`/bin/javac` with JAVAC, and all occurrences of *`JDK14_HOME`*`/bin/java` with JAVA, except where JAVA and JAVAC are defined.

4. Set the shared library path environment variable for your platform to include the `$ORACLE_HOME/lib` directory:

Platform	Environment Variable
AIX	LIBPATH
HP-UX	LD_LIBRARY_PATH
Solaris	LD_LIBRARY_PATH_64

Custom Make Files

Oracle recommends that you use the provided `demo_`*`product`*`.mk` make files to create user programs as described in the product-specific sections of this chapter. If you modify the provided make file, or if you choose to use a custom-written make file, the following restrictions apply:

- Do not modify the order of the Oracle libraries. Oracle libraries are included on the link line more than once so that all of the symbols are resolved during linking.

  The order of the Oracle libraries is essential for the following reasons:

  – Oracle libraries are mutually referential. Functions in library A call functions in library B, and functions in library B call functions in library A.

  – The HP-UX and Tru64 UNIX linkers are one-pass linkers. The AIX, Linux, and Solaris linkers are two-pass linkers.

- If you add your own library to the link line, add it to the beginning or to the end of the link line. Do not place user libraries between the Oracle libraries.
- If you choose to use a `make` utility such as `nmake` or GNU `make`, be aware of how macro and suffix processing differs from the make utility provided with the platform. Oracle make files are tested and are supported with the `make` utility for your platform.
- Oracle library names and the contents of Oracle libraries are subject to change between releases. Always use the `demo_`*`product`*`.mk` make file that ships with the current release as a guide to determine the required libraries.

Multi-threaded Applications

The Oracle libraries provided with this release are thread safe, allowing support for multi-threaded applications.

Using Signal Handlers

Oracle Database uses signals for two-task communication. Signals are installed in a user process when it connects to the database and are removed when it disconnects.

Table 6–5 describes the signals that Oracle Database uses for two-task communication.

Table 6–5 Signals for Two-Task Communication

Signal	Description
SIGCLD	The pipe driver uses SIGCLD, also referred to as SIGCHLD, when an Oracle process terminates. The UNIX kernel sends a SIGCLD signal to the user process. The signal handler uses the wait() routine to determine whether a server process died. The Oracle process does not catch SIGCLD; the user process catches it.
SIGCONT	The pipe two-task driver uses SIGCONT to send out-of-band breaks from the user process to the Oracle process.
SIGINT	Two-task drivers use SIGINT to detect user interrupt requests. The Oracle process does not catch SIGINT; the user process catches it.
SIGIO	Oracle Net protocols use SIGIO to indicate incoming networking events.
SIGPIPE	The pipe driver uses SIGPIPE to detect end-of-file on the communications channel. When writing to the pipe, if no reading process exists, a SIGPIPE signal is sent to the writing process. Both the Oracle process and the user process catch SIGPIPE. SIGCLD is similar to SIGPIPE, but it applies only to user processes, not to Oracle processes.
SIGTERM	The pipe driver uses SIGTERM to signal interrupts from the user to the Oracle process. This occurs when the user presses the interrupt key, Ctrl+c. The user process does not catch SIGTERM; the Oracle process catches it.
SIGURG	Oracle Net TCP/IP drivers use SIGURG to send out-of-band breaks from the user process to the Oracle process.

The listed signals affect all precompiler applications. You can install one signal handler for SIGCLD (or SIGCHLD) and SIGPIPE when connected to the Oracle process. If you call the osnsui() routine to set it up, you can have more than one

signal handle for SIGINT. For SIGINT, use osnsui() and osncui() to register and delete signal-catching routines.

You can also install as many signal handlers as you want for other signals. If you are not connected to the Oracle process, you can have multiple signal handlers.

Example 6–1 shows how to set up a signal routine and a catching routine.

Example 6–1 Signal Routine and Catching Routine

```
/* user side interrupt set */
word osnsui( /*_ word *handlp, void (*astp), char * ctx, _*/)
/*
** osnsui: Operating System dependent Network Set User-side Interrupt. Add an
** interrupt handling procedure astp. Whenever a user interrupt(such as a ^C)
** occurs, call astp with argument ctx. Put in *handlp handle for this
** handler so that it may be cleared with osncui. Note that there may be many
** handlers; each should be cleared using osncui. An error code is returned if
** an error occurs.
*/

/* user side interrupt clear */
word osncui( /*_ word handle _*/ );
/*
** osncui: Operating System dependent Clear User-side Interrupt. Clear the
** specified handler. The argument is the handle obtained from osnsui. An error
** code is returned if an error occurs.
*/
```

Example 6–2 shows how to use the osnsui() and the osncui() routines in an application program.

Example 6–2 osnsui() and osncui() Routine Template

```
/*
** User interrupt handler template.
*/
void sig_handler()
{
...
}

main(argc, argv)
int arc;
char **argv;
{

        int handle, err;
        ...

        /* set up my user interrupt handler */

        if (err = osnsui(&handle, sig_handler, (char *) 0))
        {
                /* if the return value is non-zero, an error has occurred
                   Take appropriate action for the error. */
                ...
        }

        ...

        /* clear my interrupt handler */

        if (err = osncui(handle))
        {
                /* if the return value is non-zero, an error has occurred
                   Take appropriate action for the error. */
                ...
        }
        ...
}
```

XA Functionality

Oracle XA is the Oracle implementation of the X/Open Distributed Transaction Processing (DTP) XA interface. The XA standard specifies a bidirectional interface between resource managers (for example, Oracle) that provide access to shared resources within transactions, and between a transaction service that monitors and resolves transactions.

Oracle Call Interface has XA functionality. When building a TP-monitor XA application, ensure that the TP-monitor libraries (that define the symbols ax_reg and ax_unreg) are placed in the link line before the Oracle client shared library. This link restriction is required only when using the XA dynamic registration (Oracle XA switch `xaoswd`).

Oracle Database XA calls are defined in both the client shared library (`libclntsh.a`, `libclntsh.sl`, or `libclntsh.so` depending on your platform) and the client static library (`libclntst.a`). These libraries are located in the `$ORACLE_HOME/lib` directory.

7

SQL*Loader and PL/SQL Demonstrations

This chapter describes how to build and run the SQL*Loader and PL/SQL demonstration programs available with Oracle Database 10*g*. It contains the following sections:

- SQL*Loader Demonstrations
- PL/SQL Demonstrations
- Calling 32-Bit External Procedures from PL/SQL

Note: To use the demonstrations described in this chapter, you must install the Oracle Database Examples included on the Oracle Database 10*g* Companion CD.

You must also unlock the SCOTT account and set the password before creating the demonstrations.

SQL*Loader Demonstrations

Run the `ulcase.sh` file to run all of the SQL*Loader demonstrations. To run an individual demonstration, read the information contained in the file to determine how to run it.

PL/SQL Demonstrations

PL/SQL includes a number of demonstration programs that you can load. Oracle Database 10*g* must be open and mounted to work with the demonstration programs.

You must build database objects and load sample data before using these programs. To build the objects and load the sample data:

1. Change directory to the PL/SQL demonstrations directory:

   ```
   $ cd $ORACLE_HOME/plsql/demo
   ```

2. Start SQL*Plus and connect as SCOTT/TIGER:

   ```
   $ sqlplus SCOTT/TIGER
   ```

3. Enter the following commands to build the objects and load the sample data:

   ```
   SQL> @exampbld.sql
   SQL> @examplod.sql
   ```

> **Note:** Build the demonstrations as any Oracle user with sufficient privileges. Run the demonstrations as the same Oracle user.

PL/SQL Kernel Demonstrations

The following PL/SQL kernel demonstrations are available:

- `examp1.sql`
- `examp2.sql`
- `examp3.sql`
- `examp4.sql`
- `examp5.sql`
- `examp6.sql`

- examp7.sql
- examp8.sql
- examp11.sql
- examp12.sql
- examp13.sql
- examp14.sql
- sample1.sql
- sample2.sql
- sample3.sql
- sample4.sql
- extproc.sql

To compile and run the examp*n*.sql or sample*n*.sql PL/SQL kernel demonstrations:

1. Start SQL*Plus and connect as SCOTT/TIGER:

```
$ cd $ORACLE_HOME/plsql/demo
$ sqlplus SCOTT/TIGER
```

2. Enter a command similar to the following to run a demonstration, where *demo_name* is the name of the demonstration:

```
SQL> @demo_name
```

To run the extproc.sql demonstration, follow these steps:

1. If necessary, add an entry for external procedures to the tnsnames.ora file, similar to the following:

```
EXTPROC_CONNECTION_DATA =
   (DESCRIPTION =
       (ADDRESS_LIST =
          (ADDRESS=(PROTOCOL = IPC)( KEY = EXTPROC))
       )
       (CONNECT_DATA =
          (SID = PLSExtProc)
       )
    )
```

2. If necessary, add an entry for external procedures to the `listener.ora` file, similar to the following, depending on your operating system:

   Note: The value that you specify for SID_NAME in the `listener.ora` file must match the value that you specify for SID in the `tnsnames.ora` file.

   - HP-UX, Linux, Solaris, and Tru64 UNIX:

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC=
        (SID_NAME=PLSExtProc)
        (ORACLE_HOME=oracle_home_path)
        (ENVS=EXTPROC_DLLS=oracle_home_path/plsql/demo/extproc.so,
           LD_LIBRARY_PATH=oracle_home_path/plsql/demo)
        (PROGRAM=extproc)
      )
    )
```

   - AIX:

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC=
        (SID_NAME=PLSExtProc)
        (ORACLE_HOME=oracle_home_path)
        (ENVS=EXTPROC_DLLS=oracle_home_path/plsql/demo/extproc.so,
           LIBPATH=oracle_home_path/plsql/demo)
        (PROGRAM=extproc)
      )
    )
```

3. Change directory to `$ORACLE_HOME/plsql/demo`.

4. Enter the following command to create the `extproc.so` shared object, build the required database objects, and load the sample data:

```
$ make -f demo_plsql.mk extproc.so exampbld examplod
```

   Alternatively, if you have already built the database objects and loaded the sample data, enter the following command:

```
$ make -f demo_plsql.mk extproc.so
```

5. From SQL*Plus, enter the following commands:

```
SQL> CONNECT SYSTEM/SYSTEM_password
SQL> GRANT CREATE LIBRARY TO SCOTT;
SQL> CONNECT SCOTT/TIGER
SQL> CREATE OR REPLACE LIBRARY demolib IS
  2  'oracle_home_path/plsql/demo/extproc.so';
  3  /
```

6. To run the demonstration, enter the following command:

```
SQL> @extproc
```

PL/SQL Precompiler Demonstrations

> **Note:** The `make` commands shown in this section build the required database objects and load the sample data in the SCOTT schema.

The following precompiler demonstrations are available:

- `examp9.pc`
- `examp10.pc`
- `sample5.pc`
- `sample6.pc`

To build all of the PL/SQL precompiler demonstrations, set the library path environment variable to include the `$ORACLE_HOME/lib` directory, and enter the following commands:

```
$ cd $ORACLE_HOME/plsql/demo
$ make -f demo_plsql.mk demos
```

To build a single demonstration, enter its name as the argument in the `make` command. For example, to build the `examp9` demonstration, enter:

```
$ make -f demo_plsql.mk examp9
```

To run the `examp9` demonstration, enter the following command:

```
$ ./examp9
```

Calling 32-Bit External Procedures from PL/SQL

Note: This section applies to AIX, HP-UX, and Solaris only.

The 64-bit external procedure executable (`extproc`) and the 32-bit external procedure executable (`extproc32`) are installed in the `$ORACLE_HOME/bin` directory. By default, the `extproc` executable is enabled to run 64-bit external procedures on AIX, HP-UX, and Solaris. To enable 32-bit external procedures:

1. Set the value of the PROGRAM parameter in the `listener.ora` file:

   ```
   (PROGRAM=extproc32)
   ```

2. Include the `$ORACLE_HOME/lib32` directory in one of the following environment variables, depending on your platform:

Platform	Environment Variable
AIX	LIBPATH
HP-UX	SHLIB_PATH
Solaris	LD_LIBRARY_PATH

3. Restart the listener.

Note: You can configure a listener to run either 32-bit or 64-bit external procedures, but not both at the same time. However, you can configure two listeners if you need to support both 32-bit and 64-bit external procedures.

8

Tuning for Oracle Database on UNIX

This chapter describes how to tune Oracle Database on UNIX systems. It contains the following sections:

Importance of Tuning

Oracle Database is a highly optimizable software product. Frequent tuning optimizes system performance and prevents data bottlenecks. Although this chapter is written from the perspective of single-node systems, most of the performance tuning tips provided here also apply to Oracle Real Application Clusters installations.

Before tuning the system, observe its normal behavior using the tools described in the "Operating System Tools" section on page 8-2.

Operating System Tools

Several operating system tools are available to help you assess database performance and determine database requirements. In addition to providing statistics for Oracle processes, these tools provide statistics for CPU usage, interrupts, swapping, paging, context switching, and I/O for the entire system.

Common Tools

The following sections provide information on common tools:

- vmstat
- sar
- iostat
- swap, swapinfo, swapon, or lsps

> **See Also:** For more information about these tools, see the operating system documentation and UNIX man pages.

vmstat

Use the `vmstat` command to view process, virtual memory, disk, trap, and CPU activity, depending on the switches that you supply with the command. Enter one of the following commands to display a summary of CPU activity six times, at five-second intervals:

- HP-UX and Solaris:

```
$ vmstat -S 5 6
```

- AIX, Linux, and Tru64 UNIX:

```
$ vmstat 5 6
```

The following example shows output from the command on Solaris:

```
procs     memory            page            disk          faults      cpu
 r b w   swap  free  si  so pi po fr de sr f0 s0 s1 s3   in   sy   cs us sy id
 0 0 0   1892  5864   0   0  0  0  0  0  0  0  0  0  0   90   74   24  0  0 99
 0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   46   25   21  0  0 100
 0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   47   20   18  0  0 100
 0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  2   53   22   20  0  0 100
 0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   87   23   21  0  0 100
 0 0 0  85356  8372   0   0  0  0  0  0  0  0  0  0  0   48   41   23  0  0 100
```

The `w` column, under the `procs` column, shows the number of potential processes that have been swapped out and written to disk. If the value is not zero, swapping is occurring and your system is short of memory.

The `si` and `so` columns under the `page` column indicate the number of swap-ins and swap-outs per second, respectively. Swap-ins and swap-outs should always be zero.

The `sr` column under the `page` column indicates the scan rate. High scan rates are caused by a shortage of available memory.

The `pi` and `po` columns under the `page` column indicate the number of page-ins and page-outs per second, respectively. It is normal for the number of page-ins and page-outs to increase. Some paging always occurs even on systems with lots of available memory.

> **Note:** The output from the `vmstat` command differs between platforms. See the man page for information about interpreting the output on your platform.

sar

Use the `sar` (system activity reporter) command to display cumulative activity counters in the operating system, depending on the switches that you supply with the command. The following command displays a summary of I/O activity ten times, at ten-second intervals:

```
$ sar -b 10 10
```

The following example shows output from the command on Solaris:

```
13:32:45 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
13:32:55       0      14     100       3      10      69       0       0
13:33:05       0      12     100       4       4       5       0       0
13:33:15       0       1     100       0       0       0       0       0
13:33:25       0       1     100       0       0       0       0       0
13:33:35       0      17     100       5       6       7       0       0
13:33:45       0       1     100       0       0       0       0       0
13:33:55       0       9     100       2       8      80       0       0
13:34:05       0      10     100       4       4       5       0       0
13:34:15       0       7     100       2       2       0       0       0
13:34:25       0       0     100       0       0     100       0       0

Average        0       7     100       2       4      41       0       0
```

Note: On Tru64 UNIX systems, the `sar` command is available in the UNIX SVID2 compatibility subset, OSFSVID.

The `sar` output provides a snapshot of system I/O activity at a point in time. If you specify the interval time with more than one option, the output can become difficult to read. If you specify an interval time of less than 5, the `sar` activity itself can affect the output. For more information about `sar`, refer to the man page.

iostat

Use the `iostat` command to view terminal and disk activity, depending on the switches that you supply with the command. The output from the `iostat` command does not include disk request queues, but it shows which disks are busy. This information is valuable when you need to balance I/O loads.

The following command displays terminal and disk activity five times, at five-second intervals:

```
$ iostat 5 5
```

The following example shows output from the command on Solaris:

```
tty          fd0           sd0           sd1           sd3          cpu
 tin tout Kps tps serv  Kps tps serv  Kps tps serv  Kps tps serv  us sy wt id
   0    1   0   0    0    0   0   31    0   0   18    3   0   42   0  0  0 99
   0   16   0   0    0    0   0    0    0   0    0    1   0   14   0  0  0 100
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
   0   16   0   0    0    0   0    0    2   0   14   12   2   47   0  0  1 98
```

Use the `iostat` command to look for large disk request queues. A request queue shows how long the I/O requests on a particular disk device must wait to be serviced. Request queues are caused by a high volume of I/O requests to that disk or by I/O with long average seek times. Ideally, disk request queues should be at or near zero.

swap, swapinfo, swapon, or lsps

Use the `swap`, `swapinfo`, `swapon`, or `lsps` command to report information about swap space usage. A shortage of swap space can stop processes responding, leading to process failures with 'Out of Memory' errors. The following table lists the appropriate command to use for each platform:

Platform	Command
AIX	`lsps -a`
HP-UX	`swapinfo -m`
Linux	`swapon -s`
Solaris	`swap -l` and `swap -s`
Tru64 UNIX	`swapon -s`

The following example shows sample output from the `swap -l` command on Solaris:

```
swapfile             dev        swaplo blocks        free
/dev/dsk/c0t3d0s1    32,25      8      197592        162136
```

AIX Tools

The following sections describe tools available on AIX systems.

> **See Also:** For more information about these tools, see the AIX operating system documentation and man pages.

AIX System Management Interface Tool

The AIX System Management Interface Tool (SMIT) provides a menu-driven interface to various system administrative and performance tools. Using SMIT, you can navigate through large numbers of tools and focus on the jobs that you want to perform.

Base Operation System Tools

The AIX Base Operation System (BOS) contains performance tools that are historically part of UNIX systems or are required to manage the implementation-specific features of AIX. The following table lists the most important BOS tools:

Tool	Description
`lsattr`	Displays the attributes of devices
`lslv`	Displays information about a logical volume or the logical volume allocations of a physical volume
`netstat`	Displays the contents of network-related data structures
`nfsstat`	Displays statistics about Network File System (NFS) and Remote Procedure Call (RPC) activity
`nice`	Changes the initial priority of a process
`no`	Displays or sets network options
`ps`	Displays the status of one or more processes
`reorgvg`	Reorganizes the physical-partition allocation within a volume group
`time`	Displays the elapsed execution, user CPU processing, and system CPU processing time
`trace`	Records and reports selected system events
`vmo`	Manages Virtual Memory Manager tunable parameters

AIX Performance Toolbox

The AIX Performance Toolbox (PTX) contains tools for monitoring and tuning system activity locally and remotely. PTX consists of two main components, the PTX Manager and the PTX Agent. The PTX Manager collects and displays data from various systems in the configuration by using the `xmperf` utility. The PTX Agent collects and transmits data to the PTX Manager by using the `xmserd` daemon. The PTX Agent is also available as a separate product called Performance Aide for AIX.

Both PTX and Performance Aide include the following monitoring and tuning tools:

Tool	Description
`fdpr`	Optimizes an executable program for a particular workload
`filemon`	Uses the trace facility to monitor and report the activity of the file system
`fileplace`	Displays the placement of a file's blocks within logical or physical volumes
`lockstat`	Displays statistics about contention for kernel locks
`lvedit`	Facilitates interactive placement of logical volumes within a volume group
`netpmon`	Uses the trace facility to report on network I/O and network-related CPU usage
`rmss`	Simulates systems with various memory sizes for performance testing
`svmon`	Captures and analyzes information about virtual-memory usage
`syscalls`	Records and counts system calls
`tprof`	Uses the trace facility to report CPU usage at module and source-code-statement levels
`BigFoot`	Reports the memory access patterns of processes
`stem`	Permits subroutine-level entry and exit instrumentation of existing executables

See Also: For more information about these tools, see the *Performance Toolbox for AIX Guide and Reference*, and for more information about the syntax of some of these tools, see the *AIX 5L Performance Management Guide*.

HP-UX Tools

The following sections describe tools available on HP-UX systems.

Performance Tuning Tools

The following table lists the tools that you can use for additional performance tuning on HP-UX:

> **See Also:** For more information about these tools, see the HP-UX operating system documentation and man pages.

Tools	Description
`caliper` (Itanium only)	Collects run-time application data for system analysis tasks such as cache misses, translation look-aside buffer (TLB) or instruction cycles, along with fast dynamic instrumentation. It is a dynamic performance measurement tool for C, C++, Fortran, and assembly applications.
`gprof`	Creates an execution profile for programs.
`monitor`	Monitors the program counter and calls to certain functions.
`netfmt`	Monitors the network.
`netstat`	Reports statistics on network performance.
`nfsstat`	Displays statistics about Network File System (NFS) and Remote Procedure Call (RPC) activity.
`nettl`	Captures network events or packets by logging and tracing.
`prof`	Creates an execution profile of C programs and displays performance statistics for your program, showing where your program is spending most of its execution time.
`profil`	Copies program counter information into a buffer.
`top`	Displays the top processes on the system and periodically updates the information.

HP-UX Performance Analysis Tools

The following HP-UX performance analysis tools are also available on HP-UX systems:

- GlancePlus/UX
- HP PAK

GlancePlus/UX

This HP-UX utility is an online diagnostic tool that measures the system's activities. GlancePlus displays how system resources are being used. It displays dynamic information about the system's I/O, CPU, and memory usage in a series of screens. You can also use the utility to monitor how individual processes are using resources.

HP PAK

HP Programmer's Analysis Kit (HP PAK) currently consists of two tools, Puma and Thread Trace Visualizer (TTV):

- Puma collects performance statistics during a program run. It provides several graphical displays for viewing and analyzing the collected statistics.
- TTV displays trace files produced by the instrumented thread library, `libpthread_tr.sl`, in a graphical format. It allows you to view how threads are interacting and to find where threads are blocked waiting for resources.

HP PAK is bundled with the HP Fortran 77, HP Fortran 90, HP C, HP C++, HP ANSI C++, and HP Pascal compilers.

Linux Tools

On Linux systems, use the `top`, `free`, and `cat /proc/meminfo` command to view information about swap space, memory, and buffer usage.

Solaris Tools

On Solaris systems, use the `mpstat` command to view statistics for each processor in a multiprocessor system. Each row of the table represents the activity of one processor. The first row summarizes all activity since the last system reboot; each subsequent row summarizes activity for the preceding interval. All values are events per second unless otherwise noted. The arguments are for time intervals between statistics and number of iterations. The following example shows sample output from the `mpstat` command:

```
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0    1    71   21   23    0    0    0    0    55    0   0   0  99
  2    0   0    1    71   21   22    0    0    0    0    54    0   0   0  99
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0    0    61   16   25    0    0    0    0    57    0   0   0 100
  2    1   0    0    72   16   24    0    0    0    0    59    0   0   0 100
```

Tuning Memory Management

Start the memory tuning process by measuring paging and swapping space to determine how much memory is available. After you have determined your system memory usage, tune the Oracle buffer cache.

The Oracle buffer manager ensures that the more frequently accessed data is cached longer. If you monitor the buffer manager and tune the buffer cache, you can have a significant influence on Oracle Database performance. The optimal Oracle Database buffer size for your system depends on the overall system load and the relative priority of Oracle over other applications.

Allocate Sufficient Swap Space

Try to minimize swapping because it causes significant UNIX overhead. To check for swapping, use the `sar` or `vmstat` commands. For information about the appropriate options to use with these commands, see the man pages.

If your system is swapping and you must conserve memory:

- Avoid running unnecessary system daemon processes or application processes.
- Decrease the number of database buffers to free some memory.
- Decrease the number of UNIX file buffers, especially if you are using raw devices.

To determine the amount of swap space, enter one of the following commands, depending on your platform:

Platform	Command
AIX	`lsps -a`
HP-UX	`swapinfo -m`
Linux	`swapon -s`
Solaris	`swap -l` and `swap -s`
Tru64 UNIX	`swapon -s`

To add swap space to your system, enter one of the following commands, depending on your platform:

Platform	Command
AIX	`chps` or `mkps`
HP-UX	`swapon`
Linux	`swapon -a`
Solaris	`swap -a`
Tru64 UNIX	`swapon -a`

Set the swap space to between two and four times the system's physical memory. Monitor the use of swap space and increase it as required.

> **See Also:** For more information about these commands, see your operating system documentation.

Control Paging

Paging might not present as serious a problem as swapping, because an entire program does not have to be stored in memory to run. A small number of page-outs might not noticeably affect the performance of your system.

To detect excessive paging, run measurements during periods of fast response or idle time to compare against measurements from periods of slow response.

Use the `vmstat` or `sar` command to monitor paging. See the man pages or your operating system documentation for information about interpreting the results for your platform. The following columns from the output of these commands are important on Solaris:

Column	Description
`vflt/s`	Indicates the number of address translation page faults. Address translation faults occur when a process refers to a valid page not in memory.
`rclm/s`	Indicates the number of valid pages that have been reclaimed and added to the free list by page-out activity. This value should be zero.

If your system consistently has excessive page-out activity, consider the following solutions:

- Install more memory.
- Move some of the work to another system.
- Configure the SGA to use less memory.

Adjust Oracle Block Size

A UNIX system reads entire operating system blocks from the disk. If the database block size is smaller than the UNIX file system block size, I/O bandwidth is inefficient. If you set the Oracle database block size to be a multiple of the file system blocksize, you can increase performance by up to five percent.

The DB_BLOCK_SIZE initialization parameter sets the database block size. However, to change the value of this parameter, you must recreate the database.

To see the current value of the DB_BLOCK_SIZE parameter, enter the SHOW PARAMETER DB_BLOCK_SIZE command in SQL*Plus.

Tuning Disk I/O

Balance I/O evenly across all available disks to reduce disk access times. For smaller databases and those not using RAID, ensure that different data files and tablespaces are distributed across the available disks.

Use Automatic Storage Management

If you choose to use Automatic Storage Management for database storage, all database I/O is balanced across all available disk devices in the ASM disk group. ASM provides the performance of raw device I/O without the inconvenience of managing raw devices.

By using ASM, you avoid the need to manually tune disk I/O.

Choose the Appropriate File System Type

Depending on your operating system, you can choose from a range of file system types. Each file system type has different characteristics which can have a

substantial impact on database performance. The following table lists common file system types available on UNIX platforms:

File System	Platform	Description
S5	AIX, HP-UX, Solaris	UNIX System V file system
UFS	AIX, HP-UX, Solaris, Tru64 UNIX	Unified file system, derived from BSD UNIX
VxFS	AIX, Solaris, HP-UX	VERITAS file system
None	All	Raw devices (no file system)
EXT2/EXT3	Linux	Extended file system for Linux
AdvFS	Tru64 UNIX	Advanced file system
CFS	Tru64 UNIX	Cluster file system
JFS/JFS2	AIX	Journaled file system
GPFS	AIX	General parallel file system
OCFS	Linux	Oracle Cluster file system

The suitability of a file system to an application is usually not documented. For example, even different implementations of the Unified file system are hard to compare. Performance differences can vary from 0 to 20 percent, depending on the file system that you choose. If you choose to use a file system:

- Make a new file system partition to ensure that the hard disk is clean and unfragmented.
- Perform a file system check on the partition before using it for database files.
- Distribute disk I/O as evenly as possible.
- If you are not using a logical volume manager or a RAID device, consider placing log files on a different file system from data files.

Monitoring Disk Performance

To monitor disk performance, use the `sar -b` and `sar -u` commands.

Table 8–1 describes the columns of the `sar -b` command output that are significant for analyzing disk performance.

Table 8–1 sar -b Output Columns

Columns	Description
`bread/s, bwrit/s`	Blocks read and blocks written per second (important for file system databases)
`pread/s, pwrit/s`	Partitions read and partitions written per second (important for raw partition database systems)

An important `sar -u` column for analyzing disk performance is `%wio`, the percentage of CPU time waiting on blocked I/O.

> **Note:** Not all Linux distributions display the `%wio` column in the output of the `sar -u` command. For detailed I/O statistics, you can use `iostat -x` command.

Key indicators are:

- The sum of the `bread`, `bwrit`, `pread`, and `pwrit` columns indicates the level of activity of the disk I/O subsystem. The higher the sum, the busier the I/O subsystem. The larger the number of physical drives, the higher the sum threshold number can be. A good default value is no more than 40 for two drives and no more than 60 for four to eight drives.
- The `%rcache` column value should be greater than 90 and the `%wcache` column value should be greater than 60. Otherwise, the system may be disk I/O bound.
- If the `%wio` column value is consistently greater than 20, the system is I/O bound.

System Global Area

The System Global Area (SGA) is the Oracle structure that is located in shared memory. It contains static data structures, locks, and data buffers. Sufficient shared memory must be available to each Oracle process to address the entire SGA.

The maximum size of a single shared memory segment is specified by the shmmax kernel parameter (shm_max on Tru64 UNIX). The following table shows the recommended value for this parameter, depending on your platform:

Platform	Recommended Value
AIX	Not applicable.
HP-UX	The size of the physical memory installed on the system. **See Also:** HP-UX Shared Memory Segments for an Oracle Instance on page B-2 for information about the shmmax parameter on HP-UX.
Linux	Half the size of the physical memory installed on the system.
Solaris	4294967295 or 4 GB minus 16 MB. Can be greater than 4 GB on 64-bit systems.
Tru64 UNIX	4294967295 or 4 GB minus 16 MB. **Note:** The value of the shm_max parameter must be at least 16 MB for the Oracle instance to start. If your system runs both Oracle9*i* and Oracle Database 10*g* instances, you must set the value of this parameter to 2 GB minus 16 MB.

If the size of the SGA exceeds the maximum size of a shared memory segment (shmmax or shm_max), Oracle Database 10*g* attempts to attach more contiguous segments to fulfill the requested SGA size. The shmseg kernel parameter (shm_seg on Tru64 UNIX) specifies the maximum number of segments that can be attached by any process. Set the following initialization parameters to control the size of the SGA:

- DB_CACHE_SIZE
- DB_BLOCK_SIZE
- JAVA_POOL_SIZE
- LARGE_POOL_SIZE
- LOG_BUFFERS
- SHARED_POOL_SIZE

Alternatively, set the SGA_TARGET initialization parameter to enable Oracle to automatically tune the SGA size.

Use caution when setting values for these parameters. When values are set too high, too much of the system's physical memory is devoted to shared memory, resulting in poor performance.

Oracle databases configured with Shared Server require a higher setting for the SHARED_POOL_SIZE initialization parameter, or a custom configuration that uses the LARGE_POOL_SIZE initialization parameter. If you installed the database with the Oracle Universal Installer, then the value of the SHARED_POOL_SIZE parameter is set automatically by Database Configuration Assistant. However, if you created a database manually, increase the value of the SHARED_POOL_SIZE parameter in the parameter file by 1 KB for each concurrent user.

Determine the Size of the SGA

You can determine the SGA size in one of the following ways:

- Enter the following SQL*Plus command to display the size of the SGA for a running database:

  ```
  SQL> SHOW SGA
  ```

  The result is shown in bytes.

- Determine the size of the SGA when you start your database instance. The SGA size is displayed next to the heading Total System Global Area.
- Enter the `ipcs` command as the `oracle` user.

Shared Memory on AIX

On AIX, shared memory uses common virtual memory resources across processes. Processes share virtual memory segments through a common set of virtual memory translation resources, for example tables and cached entries, for improved performance.

With Oracle Database on AIX, shared memory can be pinned to prevent paging and to reduce I/O overhead. To do this, set the LOCK_SGA parameter to TRUE. On AIX 5L, the same parameter activates the large page feature whenever the underlying hardware supports it.

Enter the following command to make pinned memory available to Oracle Database on AIX systems:

```
$ /usr/sbin/vmo -r -o v_pinshm=1
```

Enter a command similar to the following to set the maximum percentage of real memory available for pinned memory, where *percent_of_real_memory* is the maximum percent of real memory that you want to set:

```
$ /usr/sbin/vmo -r -o maxpin%=percent_of_real_memory
```

When using the `maxpin%` option, it is important that the amount of pinned memory exceeds the Oracle SGA size by at least 3 percent of the real memory on the system, allowing free pinnable memory for use by the kernel. For example, if you have 2 GB of physical memory and you want to pin the SGA by 400 MB (20 percent of the RAM), then enter the following command:

```
$ /usr/sbin/vmo -r -o maxpin%=23
```

Use the `svmon` command to monitor the use of pinned memory during the operation of the system. Oracle Database attempts to pin memory only if the LOCK_SGA parameter is set to TRUE.

Large Page Feature on AIX POWER4-Based Systems

To turn on and reserve 10 large pages each of size 16 MB on a POWER4 system, enter the following command:

```
$ /usr/sbin/vmo -r -o lgpg_regions=10 -o lgpg_size=16777216
```

This command proposes bosboot and warns that a reboot is required for the changes to take affect.

Oracle recommends specifying enough large pages to contain the entire SGA. The Oracle instance attempts to allocate large pages when the LOCK_SGA parameter is set to TRUE. If the SGA size exceeds the size of memory available for pinning, or large pages, the portion of the SGA exceeding these sizes is allocated to ordinary shared memory.

> **See Also:** For more information about enabling and tuning pinned memory and large pages, see the AIX documentation.

Tuning the Operating System Buffer Cache

To take full advantage of raw devices, adjust the size of the Oracle Database buffer cache and, if memory is limited, the operating system buffer cache.

The operating system buffer cache holds blocks of data in memory while they are being transferred from memory to disk, or from disk to memory.

The Oracle Database buffer cache is the area in memory that stores the Oracle database buffers. Because Oracle Database can use raw devices, it does not need to use the operating system buffer cache.

If you use raw devices, increase the size of the Oracle Database buffer cache. If the amount of memory on the system is limited, make a corresponding decrease in the operating system buffer cache size.

Use the `sar` command to determine which buffer caches you must increase or decrease. For more information about the `sar` command, see the man page.

Note: For Tru64 UNIX, do not reduce the operating system buffer cache, because the operating system automatically resizes the amount of memory that it requires for buffering file system I/O. Restricting the operating system buffer cache can cause performance issues.

A

Administering Oracle Database on AIX

This appendix contains information about administering Oracle Database on AIX. It contains the following sections:

Memory and Paging

Memory contention occurs when processes require more memory than is available. To cope with the shortage, the system pages programs and data between memory and disks.

Controlling Buffer-Cache Paging Activity

Excessive paging activity decreases performance substantially. This can become a problem with database files created on journaled file systems (JFS and JFS2). In this situation, a large number of SGA data buffers might also have analogous file system buffers containing the most frequently referenced data. The behavior of the AIX file buffer cache manager can have a significant impact on performance. It can cause an I/O bottleneck, resulting in lower overall system throughput.

On AIX, tuning buffer-cache paging activity is possible but you must do it carefully and infrequently. Use the `/usr/sbin/vmo` command to tune the following AIX system parameters:

Parameter	Description
minfree	The minimum free-list size. If the free-list space in the buffer falls below this size, the system uses page stealing to replenish the free list.
maxfree	The maximum free-list size. If the free-list space in the buffer exceeds this size, the system stops using page stealing to replenish the free list.
minperm%	Specifies the point (in percentage of total number of memory frames) below which the page-stealer will steal file or computational pages regardless of repaging rates.
maxperm%	Specifies the point (in percentage of total number of memory frames) above which the page-stealing algorithm steals only file pages.

See Also: For more information about AIX system parameters, see the *AIX 5L Performance Management Guide*.

Tuning the AIX File Buffer Cache

The purpose of the AIX file buffer cache is to reduce disk access frequency when journaled file systems are used. If this cache is too small, disk usage increases and potentially saturates one or more disks. If the cache is too large, memory is wasted.

> **See Also:** For more information about the implications of increasing the AIX file buffer cache, see "Controlling Buffer-Cache Paging Activity" on page A-2.

You can configure the AIX file buffer cache by adjusting the minperm% and maxperm% parameters. In general, if the buffer hit ratio is low (less than 90 percent), as determined by the `sar -b` command, increasing the minperm% parameter value might help. If maintaining a high buffer hit ratio is not critical, decreasing the minperm% parameter value increases the physical memory available. Refer to the AIX documentation for more information about increasing the size of the AIX file buffer cache.

The performance gain cannot be quantified easily, because it depends on the degree of multiprogramming and the I/O characteristics of the workload.

Tuning the minperm% and maxperm% Parameters

AIX provides a mechanism for you to loosely control the ratio of page frames used for files rather than those used for computational (working or program text) segments by adjusting the minperm% and maxperm% parameter values according to the following guidelines:

- If the percentage of real memory occupied by file pages falls below the minperm% value, the virtual memory manager (VMM) page-replacement algorithm steals both file and computational pages, regardless of repage rates.
- If the percentage of real memory occupied by file pages rises above the maxperm% value, the virtual memory manager page-replacement algorithm steals both file and computational pages.
- If the percentage of real memory occupied by file pages is between the minperm% and maxperm% parameter values, the virtual memory manager normally steals only file pages, but if the repaging rate for file pages is higher then the repaging rate for computational pages, the computational pages are stolen as well.

Use the following algorithm to calculate the default values:

- minperm (in pages) = ((number of page frames)-1024) * 0.2

  Then convert the minperm to a minperm% by dividing it by the total number of real-memory page frames.

- maxperm (in pages) = ((number of page frames)-1024) * 0.8

  Then convert the maxperm to a maxperm% by dividing it by the total number of real-memory page frames.

Use the following command to change the value of the minperm% parameter to 5 percent of the total number of page frames, and the value of the maxperm% parameter to 20 percent of the total number of page frames:

```
# /usr/sbin/vmo -r -o minperm%=5 -o maxperm%=20
```

The default values are 20 percent and 80 percent, respectively.

To optimize for quick response when opening new database connections, adjust the minfree parameter to maintain enough free pages in the system to load the application into memory without adding additional pages to the free list. To determine the real memory size (resident set size, working set) of a process, use the following command:

```
$ ps v process_id
```

Set the minfree parameter to this value or to 8 frames, whichever is larger.

If the database files are on raw devices, or if you are using Direct I/O, you can set the minperm% and maxperm% parameters to low values, for example 5 percent and 20 percent, respectively. This is because the AIX file buffer cache is not used either for raw devices or for Direct I/O. The memory might be better used for other purposes, such as for the Oracle System Global Area.

Allocating Sufficient Paging Space (Swap Space)

Inadequate paging space (swap space) usually causes the system to hang or suffer abnormally slow response times. On AIX, you can dynamically add paging space on raw disk partitions. The amount of paging space you should configure depends on the amount of physical memory present and the paging space requirements of your applications. Use the `lsps` command to monitor paging space use and the `vmstat` command to monitor system paging activities. To increase the paging space, use the `smit pgsp` command.

On platforms where paging space is pre-allocated, Oracle recommends that you set the paging space to a value larger than the amount of RAM. But on AIX paging space is not allocated until needed. The system uses swap space only if it runs out of real memory. If the memory is sized correctly, there is no paging and the page space can be small. Workloads where the demand for pages does not fluctuate significantly perform well with a small paging space. Workloads likely to have peak periods of increased paging require enough paging space to handle the peak number of pages.

As a general rule, an initial setting for the paging space is half the size of RAM plus 4 GB, with an upper limit of 32 GB. Monitor the paging space use with the `lsps -a` command, and increase or decrease the paging space size accordingly. The metric `%Used` in the output of `lsps -a` is typically less than 25% on a healthy system. A properly sized deployment should require very little paging space and an excessive amount of swapping is an indication that the RAM on the system might be undersized.

CAUTION: Do not undersize the paging space. If you do, the system can terminate active processes when it runs out of space. However, over-sizing the paging space has little or no negative impact.

Controlling Paging

Constant and excessive paging indicates that the real memory is over-committed. In general, you should:

- Avoid constant paging unless the system is equipped with very fast expanded storage that makes paging between memory and expanded storage much faster than Oracle can read and write data between the SGA and disks.
- Allocate limited memory resource to where it is most beneficial to system performance. It is sometimes a recursive process of balancing the memory resource requirements and trade-offs.
- If memory is not adequate, build a prioritized list of memory-requiring processes and elements of the system. Assign memory to where the performance gains are the greatest. A prioritized list might look like:
  1. OS and RDBMS kernels
  2. User and application processes
  3. Redo log buffer

4. PGAs and shared pool
5. Database block buffer caches

For instance, if you query Oracle dynamic performance tables and views and find that both the shared pool and database buffer cache require more memory, assigning the limited spare memory to the shared pool might be more beneficial than assigning it to the database block buffer caches.

The following AIX commands provide paging status and statistics:

- `vmstat -s`
- `vmstat` *`interval`* [*`repeats`*]
- `sar -r` *`interval`* [*`repeats`*]

Setting the Database Block Size

You can configure the Oracle database block size for better I/O throughput. On AIX, you can set the value of the DB_BLOCK_SIZE initialization parameter to between 2 KB and 32 KB, with a default of 4 KB. If the Oracle database is installed on a journaled file system, then the block size should be a multiple of the file system blocksize (4 KB on JFS, 16 K to 1 MB on GPFS). For databases on raw partitions, the Oracle database block size is a multiple of the operating system physical block size (512 bytes on AIX).

Oracle recommends smaller Oracle database block sizes (2 KB or 4 KB) for online transaction processing (OLTP) or mixed workload environments and larger block sizes (8 KB, 16 KB, or 32 KB) for decision support system (DSS) workload environments.

Tuning the Log Archive Buffers

By increasing the LOG_BUFFER size you might be able to improve the speed of archiving the database, particularly if transactions are long or numerous. Monitor the log file I/O activity and system throughput to determine the optimum LOG_BUFFER size. Tune the LOG_BUFFER parameter carefully to ensure that the overall performance of normal database activity does not degrade.

Note: The LOG_ARCHIVE_BUFFER_SIZE parameter was obsoleted with Oracle8*i*.

I/O Buffers and SQL*Loader

For high-speed data loading, such as using the SQL*Loader direct path option in addition to loading data in parallel, the CPU spends most of its time waiting for I/O to complete. By increasing the number of buffers, you can usually push the CPU usage harder, thereby increasing overall throughput.

The number of buffers (set by the SQL*Loader BUFFERS parameter) that you choose depends on the amount of available memory and how hard you want to push CPU usage. See *Oracle Database Utilities* for information about adjusting the file processing options string for the BUFFERS parameter.

The performance gains depend on CPU usage and the degree of parallelism that you use when loading data.

> **See Also:** For more generic information about the SQL*Loader utility, see *Oracle Database Utilities*.

BUFFER Parameter for the Import Utility

Set the BUFFER parameter for the Import utility to a large value to optimize the performance of high-speed networks when they are used. For instance, if you use the IBM RS/6000 Scalable POWERparallel Systems (SP) switch, you should set the BUFFER parameter to a value of at least 1 MB.

Disk I/O Issues

Disk I/O contention can result from poor memory management (with subsequent paging and swapping), or poor distribution of tablespaces and files across disks.

Make sure that the I/O activity is distributed evenly across multiple disk drives by using AIX utilities such as `filemon`, `sar`, `iostat`, and other performance tools to identify any disks with high I/O activity.

AIX Logical Volume Manager

The AIX Logical Volume Manager (LVM) can stripe data across multiple disks to reduce disk contention. The primary objective of striping is to achieve high performance when reading and writing large sequential files. Effective use of the striping features in the LVM allows you to spread I/O more evenly across disks, resulting in greater overall performance.

Note: Do not add logical volumes to Automatic Storage Management (ASM) disk groups. ASM works best when you add raw disk devices to disk groups. If you are using ASM, do not use LVM for striping. Automatic Storage Management implements striping and mirroring.

Design a Striped Logical Volume

When you define a striped logical volume, you must specify the following items:

Item	Recommended Settings
Drives	At least two physical drives. The drives should have minimal activity when performance-critical sequential I/O is executed. Sometimes you might need to stripe the logical volume between two or more adapters.
Stripe unit size	Although the stripe unit size can be any power of two from 2 KB to 128 KB, stripe sizes of 32 KB and 64 KB are good values for most workloads. For Oracle database files, the stripe size must be a multiple of the database block size.
Size	The number of physical partitions allocated to the logical volume must be a multiple of the number of disk drives used.
Attributes	Cannot be mirrored. Set the `copies` attribute to a value of 1.

Other Considerations

Performance gains from effective use of the LVM can vary greatly, depending on the LVM you use and the characteristics of the workload. For DSS workloads, you can see substantial improvement. For OLTP-type or mixed workloads, you can still expect significant performance gains.

Using Journaled File Systems Compared to Raw Logical Volumes

Note the following considerations when you are deciding whether to use journaled file systems or raw logical volumes:

- File systems are continually being improved, as are various file system implementations. In some cases, file systems provide better I/O performance than raw devices.
- File systems require some additional configuration (AIX minservers and maxservers parameter) and add a small CPU overhead because asynchronous I/O on file systems is serviced outside of the kernel.

- Different vendors implement the file system layer in different ways to exploit the strengths of different disks. This makes it difficult to compare file systems across platforms.
- The introduction of more powerful LVM interfaces substantially reduces the tasks of configuring and backing up logical disks based on raw logical volumes.
- The Direct I/O and Concurrent I/O feature included in AIX 5L improves file system performance to a level comparable to raw logical volumes.

If you use a journaled file system, it is easier to manage and maintain database files than if you use raw devices. In earlier versions of AIX, file systems supported only buffered read and write and added extra contention because of imperfect inode locking. These two issues are solved by the JFS2 Concurrent I/O feature and the GPFS Direct I/O feature, enabling file systems to be used instead of raw devices, even when optimal performance is required.

> **Note:** To use the Oracle Real Application Clusters option, you must place data files in an ASM disk group, on raw devices, or on a GPFS file system. You cannot use JFS or JFS2. Direct I/O is implicitly enabled when you use GPFS.

File System Options

AIX 5L includes Direct I/O and Concurrent I/O support. Direct I/O and Concurrent I/O support allows database files to exist on file systems while bypassing the operating system buffer cache and removing inode locking operations that are redundant with the features provided by Oracle Database.

Where possible, Oracle recommends enabling Concurrent I/O or Direct I/O on file systems containing Oracle data files. The following table lists file systems available on AIX and the recommended setting.

File System	Option	Description
JFS	dio	Concurrent I/O is not available on JFS. Direct I/O (dio) is available, but performance is degraded compared to JFS2 with Concurrent I/O.
JFS large file	none	Oracle does not recommend using JFS large file for Oracle Database because its 128 KB alignment constraint prevents you from using Direct I/O.

File System	Option	Description
JFS2	cio	Concurrent I/O (cio) is a better setting than Direct I/O (dio) on JFS2 because it has support for multiple concurrent readers and writers on the same file.
GPFS	N/A	Oracle Database silently enables Direct I/O on GPFS for optimum performance. GPFS' Direct I/O already supports multiple readers and writers on multiple nodes. Therefore, Direct I/O and Concurrent I/O are the same thing on GPFS.

Considerations for JFS and JFS2

Starting with Oracle Database 10*g*, you can use the FILESYSTEMIO_OPTIONS initialization parameter to configure a database to use either direct I/O or concurrent I/O when accessing datafiles, depending on the file system that is used to store them. When you specify the value SETALL for this parameter:

- Datafiles on a JFS file system are accessed using direct I/O
- Datafiles on a JFS2 file system are accessed using concurrent I/O

> **Note:** With this setting for the FILESYSTEMIO_OPTIONS parameter, you do not have to use the `cio` or `dio` mount options for these file systems to use concurrent or direct I/O. This means that you can place the software files (the Oracle home directory) and the database files on the same file system if required.

Oracle recommends that you use the value SETALL instead of the value DIRECTIO, because the DIRECTIO value disables asynchronous I/O. The default value for this parameter is ASYNC.

> **Note:** External utilities or tools cannot read datafiles on JFS2 file systems if you use the values SETALL or DIRECTIO for the FILESYSTEMIO_OPTIONS parameter. For more information, refer to the AIX documentation which covers the `open()` restriction in `cio` mode.

Although you do not have to use a separate file system for database files to use direct or concurrent I/O, you can still obtain performance benefits by placing Oracle Database log files on a separate JFS2 file system created using the

agblksize=512 option and mounted with the cio option. This delivers logging performance within a few percentage points of the performance of a raw device.

Considerations for GPFS

If you are using GPFS, you can use the same file system for all purposes including the Oracle home directory, database files, and logs. For optimal performance, use a large GPFS block size (typically at least 512 KB). GPFS is designed for scalability and there is no requirement to create multiple GPFS file systems provided that the amount of data fits in a single GPFS file system.

Moving from a Journaled File System to Raw Logical Volumes

To move from a journaled file system to raw logical volumes without having to manually reload all of the data, perform the following as the root user:

1. Create a logical volume (preferably in a big volume group) using the new raw logical volume device type (-T O), which allows putting the first Oracle block at offset zero for optimal performance:

   ```
   # mklv -T O -y new_raw_device VolumeGroup NumberOfPartitions
   ```

   Note: The raw device should be larger than the existing file. Be sure to mind the size of the new raw device to prevent wasting space.

2. Set the permissions on the raw device.
3. Use dd to convert and copy the contents of the JFS file to the new raw device, as follows:

   ```
   # dd if=old_JFS_file of=new_raw_device bs=1m
   ```

   Note: If you do not use the -T O option with the mklv command, then you must specify an offset in this dd command. Use the $ORACLE_HOME/bin/offset utility to determine the correct offset value.

4. Rename the data file.

Moving from Raw Logical Volumes to a Journaled File System

The first Oracle block on a raw logical volume is not necessarily at offset zero, whereas the first Oracle block on a file system is always at offset zero. To determine the offset and locate the first block on a raw logical volume, use the `$ORACLE_HOME/bin/offset` command. The offset can be 4096 bytes or zero on AIX logical volumes created with the `mklv -T O` option.

When you have determined the offset, you can copy data from a raw logical volume to a file system using the `dd` command and skipping the offset. The following example assumes an offset of 4096 bytes:

```
# dd if=old_raw_device bs=4k skip=1|dd of=new_file bs=256k
```

You can instruct Oracle Database to use a number of blocks smaller than the maximum capacity of a raw logical volume. If you do, you must add a `count` clause to make sure to copy only data that contains Oracle blocks. The following example assumes an offset of 4096 bytes, an Oracle block size of 8 KB, and 150000 blocks:

```
# dd if=old_raw_device bs=4k skip=1|dd bs=8k count=150000|dd of=new_file bs=256k
```

Using Asynchronous I/O

Oracle Database takes full advantage of asynchronous I/O (AIO) provided by AIX, resulting in faster database access.

AIX 5L supports asynchronous I/O (AIO) for database files created both on file system partitions and on raw devices. AIO on raw devices is implemented fully into the AIX kernel, and does not require database processes to service the AIO requests. When using AIO on file systems, the kernel database processes (`aioserver`) control each request from the time a request is taken off the queue until it completes. The kernel database processes are also used with I/O with virtual shared disks (VSDs) and HSDs with FastPath disabled. By default, FastPath is enabled. The number of `aioserver` servers determines the number of AIO requests that can be executed in the system concurrently, so it is important to tune the number of `aioserver` processes when using file systems to store Oracle Database data files.

Note: If you are using AIO with VSDs and HSDs with AIO FastPath enabled (the default), the maximum buddy buffer size must be greater than or equal to 128 KB.

Use one of the following commands to set the number of servers. This applies only when using asynchronous I/O on file systems rather than raw devices:

- `smit aio`
- `chdev -l aio0 -a maxservers='m' -a minservers='n'`

> **See Also:** For more information about SMIT, see the System Management Interface Tool (SMIT) online help, and for more information about the `smit aio` and `chdev` commands, see the man pages.

> **Note:** Starting with AIX 5L version 5.2, there are two AIO subsystems available: Legacy AIO (`aio0`) and POSIX AIO (`posix_aio0`). Oracle Database 10*g* release 1 (10.1.0) uses Legacy AIO (`aio0`). Both AIO subsystems have the same performance characteristics and the `rootpre.sh` script that you run before installing Oracle Database enables both.

Set the minimum value to the number of servers to be started at system boot. Set the maximum value to the number of servers that can be started in response to a large number of concurrent requests. These parameters apply to file systems only, they do not apply to raw devices.

The default value for the minimum number of servers is 1. The default value for the maximum number of servers is 10. These values are usually too low to run Oracle Database on large systems with 4 CPUs or more, if you are not using kernelized AIO. Oracle recommends that you set the parameters to the values listed in the following table:

Parameter	Value
maxreqs	Set the initial value to (4 * number of logical disks * queue depth). To determine the queue depth (typically 3), enter the following command: `$ lsattr -E -l hdiskxx`
minservers	Oracle recommends an initial value equal to the number of CPUs on the system or 10, whichever is lower.

Parameter	Value
maxservers	Starting with AIX 5L version 5.2, this parameter counts the maximum number of AIO servers per CPU, whereas on previous versions of AIX it was a system-wide value. If you are using GPFS, set maxservers to worker1threads divided by the number of CPUs. This is the optimal setting and increasing maxservers will not lead to additional I/O performance. If you are using JFS or JFS2, set the initial value to (10 * number of logical disks / number of CPUs) and monitor the actual number of aioservers started during a typical workload using the `pstat` or `ps` commands. If the actual number of active aioservers is equal to the maxservers, then increase the maxservers value.

If the value of the maxservers or maxreqs parameter is set too low, you will see the following warning messages repeated:

```
Warning: lio_listio returned EAGAIN
Performance degradation may be seen.
```

You can avoid these errors by increasing the value of the maxservers parameter. To display the number of AIO servers running, enter the following commands as the `root` user:

```
# pstat -a | grep -c aios
# ps -k | grep aioserver
```

Check the number of active AIO servers periodically and change the values of the minservers and maxservers parameters if necessary. The changes take place when the system restarts.

I/O Slaves

I/O Slaves are specialized Oracle processes that perform only I/O. They are rarely used on AIX, as asynchronous I/O is the default and recommended way for Oracle to perform I/O operations on AIX. I/O Slaves are allocated from shared memory buffers. I/O Slaves use a set of initialization parameters, listed in the following table.

Parameter	Range of Values	Default Value
DISK_ASYNCH_IO	TRUE/FALSE	TRUE
TAPE_ASYNCH_IO	TRUE/FALSE	TRUE
BACKUP_TAPE_IO_SLAVES	TRUE/FALSE	FALSE

Parameter	Range of Values	Default Value
DBWR_IO_SLAVES	0 - 999	0
DB_WRITER_PROCESSES	1-20	1

Generally, you do not need to adjust the parameters in the preceding table. However, on large workloads, the database writer might become a bottleneck. If it does, increase the value of the DB_WRITER_PROCESSES initialization parameter. As a general rule, do not increase the number of database writer processes above one for each 2 CPUs in the system or partition.

There are times when you need to turn off asynchronous I/O, for example, if instructed to do so by Oracle Support for debugging. You can use the DISK_ASYNCH_IO and TAPE_ASYNCH_IO initialization parameters to switch off asynchronous I/O for disk or tape devices. Because the number of I/O slaves for each process type defaults to zero, by default no I/O Slaves are deployed.

Set the DBWR_IO_SLAVES initialization parameter to a value greater than 0 only if the DISK_ASYNCH_IO or TAPE_ASYNCH_IO initialization parameter is set to FALSE. Otherwise, the database writer process (DBWR) becomes a bottleneck. In this case, the optimal value on AIX for the DBWR_IO_SLAVES initialization parameter is 4.

Using the DB_FILE_MULTIBLOCK_READ_COUNT Initialization Parameter

If you use the FILESYSTEMIO_OPTIONS initialization parameter to implement direct I/O or concurrent I/O or if you use raw devices, the file system does not perform any read-ahead on sequential scans. The read ahead is performed by Oracle Database as specified by the DB_FILE_MULTIBLOCK_READ_COUNT initialization parameter.

Setting a large value for the DB_FILE_MULTIBLOCK_READ_COUNT initialization parameter usually yields better I/O throughput on sequential scans. On AIX, the valid range for this parameter is 1 to 512, but using a value higher than 16 usually does not provide additional performance gain.

Set this parameter so that its value when multiplied by the value of the DB_BLOCK_SIZE initialization parameter produces a number larger than the LVM stripe size. Such a setting causes more disks to be used.

Using Write Behind

The write behind feature enables the operating system to group write I/Os together up to the size of a partition. Doing this increases performance because the number of I/O operations is reduced. The file system divides each file into 16 KB partitions to increase write performance, limit the number of dirty pages in memory, and minimize disk fragmentation. The pages of a particular partition are not written to disk until the program writes the first byte of the next 16 KB partition. To set the size of the buffer for write behind to eight 16 KB partitions, enter the following command:

```
# /usr/sbin/ioo -o numclust=8
```

To disable write behind, enter the following command:

```
# /usr/sbin/ioo -o numclust=0
```

Tuning Sequential Read Ahead

The information in this section applies only to file systems, and only when neither Direct I/O nor Concurrent I/O are used.

The Virtual Memory Manager (VMM) anticipates the need for pages of a sequential file. It observes the pattern in which a process accesses a file. When the process accesses two successive pages of the file, the VMM assumes that the program will continue to access the file sequentially, and schedules additional sequential reads of the file. These reads overlap the program processing and make data available to the program sooner. Two VMM thresholds, implemented as kernel parameters, determine the number of pages it reads ahead:

- minpgahead

  The number of pages read ahead when the VMM first detects the sequential access pattern

- maxpgahead

  The maximum number of pages that VMM reads ahead in a sequential file

Set the minpgahead and maxpgahead parameters to appropriate values for your application. The default values are 2 and 8 respectively. Use the `/usr/sbin/ioo` command to change these values. You can use higher values for the maxpgahead parameter in systems where the sequential performance of striped logical volumes is of paramount importance. To set the minpgahead parameter to 32 pages and the maxpgahead parameter to 64 pages, enter the following command as the `root` user:

```
# /usr/sbin/ioo –r –o minpgahead=32 –o maxpgahead=64
```

Set both the minpgahead and maxpgahead parameters to a power of two. For example, 2, 4, 8,...512, 1042... and so on.

Tuning Disk I/O Pacing

Disk I/O pacing is an AIX mechanism that allows the system administrator to limit the number of pending I/O requests to a file. This prevents disk I/O intensive processes from saturating the CPU. Therefore, the response time of interactive and CPU-intensive processes does not deteriorate.

You can achieve disk I/O pacing by adjusting two system parameters: the high-water mark and the low-water mark. When a process writes to a file that already has a pending high-water mark I/O request, the process is put to sleep. The process wakes up when the number of outstanding I/O requests falls below or equals the low-water mark.

You can use the `smit` command to change the high and low-water marks. Determine the water marks through trial-and-error. Use caution when setting the water marks because they affect performance. Tuning the high and low-water marks has less effect on disk I/O larger than 4 KB.

You can determine disk I/O saturation by analyzing the result of `iostat`, in particular, the percentage of `iowait` and `tm_act`. A high `iowait` percentage combined with high `tm_act` percentages on specific disks is an indication of disk saturation. Note that a high `iowait` alone is not necessarily an indication of an I/O bottleneck.

Minimizing Remote I/O Operations

Oracle Real Application Clusters running on the SP architecture uses VSDs or HSDs as the common storage that is accessible from all instances on different nodes. If an I/O request is to a VSD where the logical volume is local to the node, local I/O is performed. The I/O traffic to VSDs that are not local goes through network communication layers.

For better performance, it is important to minimize remote I/O as much as possible. Redo logs of each instance should be placed on the VSDs that are on local logical volumes. Each instance should have its own undo segments that are on VSDs mapped to local logical volumes if updates and insertions are intensive.

In each session, each user is allowed only one temporary tablespace. The temporary tablespaces should each contain at least one data file local to each of the nodes.

Carefully design applications and databases (by partitioning applications and databases, for instance) to minimize remote I/O.

Resilvering with Oracle Database

If you disable mirror write consistency (MWC) for an Oracle datafile allocated on a raw logical volume (LV), the Oracle Database crash recovery process uses resilvering to recover after a system crash. This resilvering process prevents database inconsistencies or corruption.

During crash recovery, if a datafile is allocated on a logical volume with more than one copy, the resilvering process performs a checksum on the data blocks of all of the copies. It then performs one of the following:

- If the data blocks in a copy have valid checksums, the resilvering process uses that copy to update the copies that have invalid checksums.
- If all copies have blocks with invalid checksums, the resilvering process rebuilds the blocks using information from the redo log file. It then writes the datafile to the logical volume and updates all of the copies.

On AIX, the resilvering process works only for datafiles allocated on raw logical volumes for which MWC is disabled. Resilvering is not required for datafiles on mirrored logical volumes with MWC enabled, because MWC ensures that all copies are synchronized.

If the system crashes while you are upgrading a previous release of Oracle Database that used datafiles on logical volumes for which MWC was disabled, enter the `syncvg` command to synchronize the mirrored LV before starting Oracle Database. If you do not synchronize the mirrored LV before starting the database, Oracle Database might read incorrect data from an LV copy.

Note: If a disk drive fails, resilvering does not occur. You must enter the `syncvg` command before you can reactivate the LV.

Caution: Oracle supports resilvering for datafiles only. Do not disable MWC for redo log files.

CPU Scheduling and Process Priorities

The CPU is another system component for which processes might contend. Although the AIX kernel allocates CPU effectively most of the time, many processes compete for CPU cycles. If your system has more than one CPU (SMP), there might be different levels of contention on each CPU.

Changing Process Running Time Slice

The default value for the runtime slice of the AIX RR dispatcher is ten milliseconds. Use the `schedtune` command to change the time slice. However, be careful when using this command. A longer time slice causes a lower context switch rate if the applications' average voluntary switch rate is lower. As a result, fewer CPU cycles are spent on context-switching for a process and the system throughput should improve.

However, a longer runtime slice can deteriorate response time, especially on a uniprocessor system. The default runtime slice is usually acceptable for most applications. When the run queue is high and most of the applications and Oracle shadow processes are capable of running a much longer duration, you might want to increase the time slice by entering the following command:

```
# /usr/sbin/schedo -o timeslice=n
```

In this command, choosing a value for *n* of 0 results in a slice of 10 milliseconds (ms), choosing a value of 1 results in a slice of 20 ms, choosing a value of 2 results in a slice of 30 ms, and so on. The time slice is calculated using the following formula:

```
(n+1) * 10
```

Using Processor Binding on SMP Systems

Binding certain processes to a processor can improve performance substantially on an SMP system. Processor binding is available and fully functional on AIX 5L.

However, starting with AIX 5L version 5.2, specific improvements in the AIX scheduler allow Oracle Database processes to be scheduled optimally without the need for processor binding. Therefore, Oracle no longer recommends binding processes to processors when running on AIX 5L version 5.2 or later.

Backing Up Raw Devices

Oracle recommends that you use Recovery Manager (RMAN) to back up raw devices. If you do use the `dd` command to back up raw devices, use it with caution, as described in this section.

The offset of the first Oracle block on a raw device may be 0 or 4 KB depending on the device subtype. You can use the `offset` command to determine the proper offset.

When creating a logical volume, Oracle recommends using an offset of zero, which is possible if you use the `-T O` option. However, existing raw logical volumes created with earlier versions of Oracle Database typically have a non-zero offset. Use the `$ORACLE_HOME/bin/offset` command to determine the offset of a logical volume containing an Oracle datafile. The following example shows how to back up and restore a raw device whose first Oracle block is at offset 4 KB:

```
$ dd if=/dev/raw_device of=/dev/rmt0.1 bs=256k
```

To restore the raw device from tape, enter commands similar to the following:

```
$ dd if=/dev/rmt0.1 of=/dev/raw_device count=63 seek=1 skip=1 bs=4k
$ mt -f /dev/rmt0.1 bsf 1
$ dd if=/dev/rmt0.1 of=/dev/raw_device seek=1 skip=1 bs=256k
```

Oracle Real Application Clusters

The following sections provide information about Oracle Real Application Clusters.

UDP Tuning

If you are using an IP-based interconnect, such as Gigabit Ethernet, Oracle Real Application Clusters uses user datagram protocol (UDP) for interprocess communications on AIX. You can tune UDP kernel parameters to improve Oracle performance. You can modify kernel UDP buffering on AIX by changing the udp_sendspace and udp_recvspace parameters. The udp_sendspace value must always be greater than the value of the Oracle Database DB_BLOCK_SIZE initialization parameter. Otherwise, one or more of the Oracle Real Application Clusters instances will fail at startup.

- Set the value of the udp_sendspace parameter to the product of DB_BLOCK_SIZE by DB_FILE_MULTIBLOCK_READ_COUNT. For example, if the value of the DB_BLOCK_SIZE parameter is 16 KB and the value of the

DB_FILE_MULTIBLOCK_READ_COUNT parameter is 16, set the udp_sendspace parameter value to 256 KB (262144).

- Set the value of the udp_recvspace parameter to at least four times the value of the udp_sendspace parameter.

You can use the following commands to change the settings:

```
# /usr/sbin/no -o udp_sendspace=65536
# /usr/sbin/no -o udp_recvspace=655360
```

To make the changes persist after system restart, edit the `/etc/rc.net` file. The following is an example of the entry in the `rc.net` file:

```
if [ -f /usr/sbin/no ] ; then
   /usr/sbin/no -o ipqmaxlen=512
   /usr/sbin/no -o udp_sendspace=65536
   /usr/sbin/no -o udp_recvspace=655360
fi
```

To monitor the suitability of the udp_recvspace parameter settings, enter the following command:

```
$ netstat -p udp | grep "socket buffer overflows"
```

If the number of overflows is not zero, increase the value of the udp_recvspace parameter. You can use the following command to reset error counters before monitoring again:

```
$ netstat -Zs -p udp
```

> **See Also:** For more information about tuning AIX parameters, see the *AIX 5L Performance Management Guide*.

Network Tuning for Transparent Application Failover

If you are experiencing a failover time of more than 10 minutes with Transparent Application Failover, consider tuning the network parameters rto_length, rto_low, and rto_high to reduce the failover time.

The lengthy Transparent Application Failover time is caused by a TCP timeout and retransmission problem in which clients connected to a crashed node do not receive acknowledgement from the failed instance. Consequently, the client continues to retransmit the same packet using an Exponential Backoff algorithm (refer to TCP/IP documentation for more information).

On AIX, the default timeout value is set to approximately 9 minutes. You can use the `no` command to tune this value using the rto_length, rto_low, and rto_high parameters. Using these parameters, you can control how often and how many times a client should retransmit the same packet before it gives up. The rto_low (default is 1 second) and rto_high (default is 64 seconds) parameters control how often to transmit the packet, while the rto_length (default is 13) parameter controls how many times to transmit the packet.

For example, using the Exponential Backoff algorithm with the AIX default values, the timeout value is set to approximately 9.3 minutes. However, using the same algorithm, and setting rto_length to 7, the timeout value is reduced to 2.5 minutes.

Note: Check the quality of the network transmission before setting any of the parameters described in this section. To check the quality of the network transmission, use the `netstat` command. Bad quality network transmissions might require a longer timeout value.

Oracle Real Application Clusters Compatibility and Clusterware

With Oracle Database 10*g*, Real Application Clusters (RAC) uses the group services provided by the AIX 5L RSCT Peer Domains (RPD). RAC no longer relies on specific services provided by HACMP. In particular, there is no need to configure the PGSD_SUBSYS variable in the information repository.

RAC is compatible with HACMP, however, HACMP is not required for RAC installations. HACMP is typically present when concurrent raw logical volumes are used instead of a GPFS file system. PSSP is not supported for this release of Oracle Database.

If you are using an IP-based interconnect, such as Gigabit Ethernet, IEEE 802.3ad, EtherChannel, or IP over SP Switch, RAC determines the interfaces to use from the IP address values specified by the CLUSTER_INTERCONNECTS initialization parameter.

Oracle Real Application Clusters and Interconnect Configuration

When you install Oracle Cluster Ready Services (CRS), you must specify the IP addresses of the private interconnects on each node. When you install RAC, it uses the private interfaces that you specified during the CRS installation. If you want to use an alternative cluster interconnect for RAC, change the value specified for the CLUSTER_INTERCONNECTS parameter.

When the interconnect (IPC) used by Real Application Clusters is based on the Internet Protocol (IP), RAC takes advantage of the fault tolerance and link aggregation provided by AIX 5L using the IEEE 802.3ad Link Aggregation or EtherChannel technologies. This replaces the Fault Tolerant IPC feature (FT-IPC) that was used in previous versions of Real Application Clusters.

Link Aggregation using 802.3ad provides the same level of fault tolerance and adds support for bandwidth aggregation. It also simplifies the configuration of Real Application Clusters.

The CLUSTER_INTERCONNECTS parameter specifies the IP addresses that RAC should use for the interconnect. This parameter typically contains only the IP address of the interface created through IEEE 802.3ad Link Aggregation or EtherChannel. For more information about EtherChannel and IEEE 802.3ad Link Aggregation, see the AIX *System Management Guide: Communications and Networks* manual.

Address Models on AIX

A 32-bit application program on AIX can be operating in one of three different virtual address models: Default Address Space Model, Large Address Space Model, and the Very Large Address Space Model. The Very Large Address-Space Model is available on AIX 5L and later versions. The address-space model in which the 32-bit application program runs determines how AIX assigns the 16 available segments, and consequently how many segments the application has available for heap and shared memory. These address space models apply only to 32-bit applications; for 64-bit applications that have access to 2^36 segments, the allocation of segments between data or shared memory is not an issue, because there are plenty of segments available. For 32-bit Oracle clients that require additional heap space, you can set the address space model according to the needs of your application.

Setting the Address Space Model

By setting the proper MAXDATA values, the address space can be split between data and shared memory as required for the 32-bit Oracle application. The following table shows examples of some of the options and their effects on Oracle memory usage:

Address Space Model	MAXDATA Value	Max Datasize	Maximum Shared Memory	Heap@	Shared Library	Comments
Default	Not specified	256 MB	2.5 GB	0x2	Shared	Default Oracle linking.
Large	0x80000000	2 GB	512 MB	0x3	Shared	This is the maximum value of MAXDATA allowed in the Large Address Space model.
Very Large	0@DSA	256 MB	3.0 GB	0x2	Private	AIX 5L, version 5.2 or later. The value 0@DSA is a special case which moves the shared library text from 0xC to the 0x2 private text segment.
Large	0x20000000	512 MB	2 GB	0x3	Shared	None.
Very Large	0x90000000@DSA	2.25 GB	256 MB	0x3	Shared	AIX 5L version 5.1 or later.

You can control the address model used by AIX by setting MAXDATA in one of the following ways:

- By setting options on the `ld` command at link time.
- By editing the header of the executable.
- By exporting the LDR_CNTRL environment variable at run time (overrides the options set using the first or second methods). If you use the LDR_CNTRL environment variable to control the memory options, the same value must be set for all Oracle processes, for example, the Oracle Net listener and SQL*Plus.

To enable the large address space model, which allows your program to have more than 256 MB of user data by allocating it from the shared memory segments, you would normally use the `-bmaxdata` link flag. For example, the `-bmaxdata:0x80000000` option would allow you to use eight 256 MB segments (segments 3 to 10) for user space (2 GB); this is the maximum and probably not what you would normally need to use.

It is also possible to patch an executable to enable the large address space model without relinking. The following script (obsoleted by `ldedit`) sets the MAXDATA value to 0x80000000, which is the maximum value:

```
$ /usr/bin/echo '\0200\0\0\0' | dd of=executable_file_name bs=4 count=1 \
   seek=19 conv=notrunc
```

In AIX 5L version 5.2, you can use the `ldedit` command to set the MAXDATA value. For example, enter the following command to change the MAXDATA value to 0x80000000@DSA:

```
$ /usr/ccs/bin/ldedit -bmaxdata:0x80000000/dsa a.out
```

AIX Kernel Modes

AIX 5L is the latest version of the AIX operating system from IBM. It is designed to exploit advanced 64-bit system and software architectures. A feature in AIX 5L is the option of running the kernel in either 64-bit or 32-bit mode. In AIX 5L version 5.2, the 32-bit kernel is installed by default, but there is an option to select the 64-bit kernel at installation time, or to select it after the installation. For information about changing the AIX kernel mode, refer to Oracle*MetaLink* Note:169426.1.

To run a 64 bit kernel or application, you need 64-bit hardware. To determine if your hardware supports 64-bit mode, enter the following command:

```
$ /usr/bin/getconf HARDWARE_BITMODE
```

If the system output returns the number 64, it can run 64-bit applications or the 64 bit kernel, or both.

To determine the mode of a kernel already running, enter the following command:

```
$ /usr/bin/getconf KERNEL_BITMODE
```

If the system returns the number 64, the 64-bit kernel is running.

> **Note:** Oracle Database 10*g* runs only on 64-bit hardware. However, you can install it on systems using either the 32-bit or 64-bit kernel.

In AIX 5L, the bit-mode (32-bit or 64-bit) of the application and kernel are independent. The only requirement to run 64-bit applications is that the hardware

supports 64-bit mode. You can run 64-bit applications using either the 32-bit or 64-bit kernel, or 32-bit applications using either the 32-bit or 64-bit kernel.

If the application requires the use of a kernel extension, then the bit-mode of the extension and the kernel must match. Oracle Database 10*g* does not require a kernel extension

On smaller systems, both bit-modes of the kernel provide similar performance, however, the 64-bit kernel provides enhanced functionality and scalability. The 64-bit kernel addresses bottlenecks which could limit throughput on 32-bit systems. The 64-bit kernel also improves scalability by allowing you to use larger sizes of physical memory. The 32-bit kernel is limited to 96 GB of physical memory. The 64-bit kernel is optimized for running 64-bit applications on POWER4 systems.

AIX Dynamic Logical Partitioning (DLPAR)

A logical partition (LPAR) is the division of a system's processors, memory, and I/O adapters into multiple environments, where each environment can be operated independently with its own operating system and applications. Logical partitioning on POWER4 pSeries servers uses hardware partitioning.

Dynamic Logical Partitioning (DLPAR) extends the capability of LPAR by providing the ability to logically attach and detach a managed system's resources (like processors, memory, or I/O adapters) to and from a hardware partition without its operating system being rebooted.

Oracle Database 10*g* can run in an AIX DLPAR logical partition and can take advantage of additional resources, such as processors, memory or I/O, which are dynamically added to the system. Also, if resources such as processors or memory are no longer required to meet the performance requirement of the instance, they can be removed while the Oracle instance continues to run and process transactions. When the instance starts, you must set the Oracle dynamic SGA parameters (such as SGA_MAX_SIZE) correctly, so that the SGA size can be increased dynamically to take advantage of additional memory which might be added to the DLPAR.

Each partition must have at least one processor, 256 MB memory, one I/O adapter associated with a boot device, and a network adapter.

The movement of an LPAR resource (processor, memory or I/O adapter) from one hardware partition to another is managed with Hardware Management Console (HMC). The Hardware Management Console (HMC) is a dedicated appliance connected to the managed system through a serial connection. The resource movement requests can be initiated either through selections made on a graphical user interface of the HMC, which can be accessed locally from the console itself, or

from a remote Web-based System Manager (WebSM) client. These resource requests can also be initiated by issuing HMC command line functions.

For more information about creating and managing LPARs and moving resources across partitions, refer to the *eServer Hardware Management Console for pSeries Installation and Operations Guide*, document number SA38-0590 from the following Web site:

`http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base/HW_hmc.htm`

B

Administering Oracle Database on HP-UX

This appendix contains information about administering Oracle Database on HP-UX. It contains the following sections:

HP-UX Shared Memory Segments for an Oracle Instance

When an Oracle Database instance starts, it creates memory segments by dividing the shared memory allocated for creating the Oracle Shared Global Area (SGA) by the value of the HP-UX shmmax kernel parameter. For example, if 64 GB of shared memory is allocated for a single Oracle instance and the value of the shmmax parameter is 1 GB, the Oracle Database creates 64 shared memory segments for that instance.

Performance degradation can occur when an Oracle instance creates multiple shared memory segments. This is because each shared memory segment receives a unique protection key when the Oracle Database creates the instance. The number of protection keys available depends on the system architecture, as follows:

Architecture	Number of Protection Keys
PA-RISC	6
Itanium	14

If the Oracle instance creates more shared memory segments than the number of protection keys, the HP-UX operating system displays protection key faults.

Oracle recommends that you set the shmmax parameter value to the amount of available physical memory on the system. Doing this ensures that the entire shared memory for a single Oracle instance is assigned to one shared memory segment and your instance needs only one protection key.

To display the list of active shared memory segments on the system, enter the following command:

```
$ ipcs -m
```

If Oracle Database creates more segments for the instance than the number of protection keys, increase the value of the shmmax kernel parameter.

> **See Also:** For more information about the recommended minimum kernel parameter values, see the *Oracle Database Installation Guide for UNIX Systems*.

HP-UX SCHED_NOAGE Scheduling Policy

On HP-UX, most processes use a time-sharing scheduling policy. Time sharing can have detrimental effects on Oracle performance by descheduling an Oracle process during critical operations, for example, when it is holding a latch. HP-UX has a modified scheduling policy, referred to as SCHED_NOAGE, that specifically addresses this issue. Unlike the normal time-sharing policy, a process scheduled using SCHED_NOAGE does not increase or decrease in priority, nor is it preempted.

This feature is suited to online transaction processing (OLTP) environments because OLTP environments can cause competition for critical resources. The use of the SCHED_NOAGE policy with Oracle Database can increase performance by 10 percent or more in OLTP environments.

The SCHED_NOAGE policy provides little or no performance gains in decision support (DSS) environments because there is little resource competition. Because each application and server environment is different, you should test and verify whether your environment benefits from the SCHED_NOAGE policy. When using SCHED_NOAGE, Oracle recommends that you exercise caution in assigning highest priority to Oracle processes. Assigning highest SCHED_NOAGE priority to Oracle processes can exhaust CPU resources on your system, causing other user processes to hang.

Enabling SCHED_NOAGE for Oracle Database

To allow Oracle Database to use the SCHED_NOAGE scheduling policy, the OSDBA group (typically the `dba` group), must have the RTSCHED and RTPRIO privileges to change the scheduling policy and set the priority level for Oracle processes. To give the `dba` group these privileges:

1. Log in as the `root` user.
2. Using any text editor, open the `/etc/privgroup` file, or create it if necessary.
3. Add or edit the following line, which begins with the name of the OSDBA group, specifying the privileges RTPRIO and RTSCHED that you want to grant to this group every time the system reboots:

   ```
   dba RTPRIO RTSCHED
   ```

4. Save the file and exit from the text editor.
5. Enter the following command to grant the privileges to the OSDBA group:

   ```
   # /usr/sbin/setprivgrp -f /etc/privgroup
   ```

6. Enter the following command to verify that the privileges are set correctly:

```
# /usr/sbin/getprivgrp dba
```

Add the HPUX_SCHED_NOAGE initialization parameter to the parameter file for each instance, setting the parameter to an integer value to specify process priority levels. The supported range of values is 178 to 255. Lower values represent higher priorities. If the parameter setting is out of range, Oracle Database automatically sets the parameter to a permissible value and continues with the SCHED_NOAGE policy with the new value. It also generates a message in the `alert_`*`sid`*`.log` file about the new setting. Whenever the highest priority is assigned to Oracle processes, either by the user or by automatic readjustment, Oracle Database generates a message in the `alert_`*`sid`*`.log` file warning about the possibility of exhaustion of CPU resources on the system. Oracle recommends that you set the parameter to assign the priority level you want for Oracle processes.

> **See Also:** For more information about priority policies and priority ranges, see the HP-UX documentation, the `rtsched(1)` man page, and the `rtsched(2)` man page.

Lightweight Timer Implementation

With Oracle Database 10g, you can collect run-time statistics at all times if the dynamic initialization parameter STATISTICS_LEVEL is set to TYPICAL (default) or ALL. This parameter setting implicitly sets the TIMED_STATISTICS initialization parameter to TRUE. Oracle Database on HP-UX systems uses the `gethrtime()` system library call to calculate elapsed times during the collection of the statistics. The use of this lightweight system library call allows you to collect run-time statistics at all times while running an Oracle instance, without affecting performance.

This system library call can provide a performance improvement of up to 10 percent over an Oracle release that does not use the `gethrtime()` system library call when the TIMED_STATISTICS initialization parameter is explicitly set to TRUE. In addition, there is no negative impact on the OLTP performance of an Oracle Database while using the `gethrtime()` system library call to collect run-time statistics at all times.

Asynchronous I/O

The asynchronous I/O pseudo-driver on HP-UX allows Oracle Database to perform I/O to raw disk partitions using an asynchronous method, resulting in less I/O overhead and higher throughput. You can use the asynchronous I/O pseudo-driver on both HP-UX servers and workstations.

MLOCK Privilege

To allow Oracle Database to execute asynchronous I/O operations, the OSDBA group (`dba`) must have the MLOCK privilege. To give the `dba` group the MLOCK privilege:

1. Log in as the `root` user.
2. Using any text editor, open the `/etc/privgroup` file, or create it if necessary.
3. Add or edit the following line, which begins with the name of the OSDBA group, specifying the privilege MLOCK:

   Note: You must use only one line to specify the privileges for a particular group in this file. If the file already contains a line for the `dba` group, add the MLOCK privilege on the same line.

   ```
   dba RTPRIO RTSCHED MLOCK
   ```

4. Save the file and exit from the text editor.
5. Enter the following command to grant the privileges to the OSDBA group:

   ```
   # /usr/sbin/setprivgrp -f /etc/privgroup
   ```

6. Enter the following command to verify that the privileges are set correctly:

   ```
   # /usr/sbin/getprivgrp dba
   ```

Implementing Asynchronous I/O

If you want to use asynchronous I/O on HP-UX, you must use one of the following storage options for database files:

- Raw devices (partitions or logical volumes)
- An Automatic Storage Management (ASM) disk group that uses raw partitions

> **See Also:** For more information about configuring ASM and raw logical volumes on HP-UX systems, see the *Oracle Database Installation Guide for UNIX Systems*.

Before you can implement asynchronous I/O with either storage option, you must use the System Administrator Management (SAM) utility to configure the asynchronous disk driver into the HP-UX kernel.

To add the asynchronous disk driver and configure the kernel using the SAM utility:

1. Enter the following command as the `root` user:

   ```
   # sam
   ```

2. Choose the Kernel Configuration area.
3. Choose the Drivers area.
4. Choose the asynchronous disk driver (`asyncdsk`).
5. Select Actions>Add Driver to Kernel.
6. Select List>Configurable Parameters.
7. Choose the MAX_ASYNC_PORTS parameter.
8. Select Action>Modify Configurable Parameter.
9. Specify a new value for the parameter, using the following guidelines, then choose OK.

   The MAX_ASYNC_PORTS parameter is a configurable HP-UX kernel parameter that controls the maximum number of processes that can open the `/dev/async` file simultaneously.

   The system displays an error when a process tries to open the `/dev/async` file after the maximum number of processes have opened the file. This error can reduce performance on systems with a large number of shadow processes or many parallel query slaves performing asynchronous I/O. This error is not recorded. To avoid this error, estimate the highest likely number of processes that can access the `/dev/async` file and set the MAX_ASYNC_PORTS parameter to this value.

10. Choose Actions>Process a New Kernel.

11. Select one of the following options, then choose OK:
    - Move Kernel Into Place and Shutdown System/Reboot Now
    - Do Not Move Kernel Into Place: Do Not Shutdown/Reboot Now

    If you choose the second option, the new kernel, `vmunix_test`, and the `system.SAM` configuration file used to create it, are both created in the `/stand/build` directory.

To use the new kernel:

1. Enter the following command to move the new kernel into place:

   ```
   # /usr/sbin/kmupdate
   ```

2. Enter the following command to reboot the system:

   ```
   # /sbin/shutdown -r now
   ```

To enable asynchronous I/O operations using the HP-UX asynchronous device driver:

1. Log in as the `root` user.
2. Enter the following command to create a new device file:

   ```
   # /sbin/mknod /dev/async c 101 0x0
   ```

3. Enter the following command to verify that the `/dev/async` device file exists and has the major number 101:

   ```
   # ls -l /dev/async
   ```

   The output of this command should look similar to the following:

   ```
   crw------- 1 oracle   dba     101 0x000000 Oct 28 10:32  /dev/async
   ```

4. If necessary, give the device file the UNIX owner and permissions consistent with those of the Oracle software owner and OSDBA group.

   If the Oracle software owner is `oracle` and the OSDBA group is `dba`, enter the following commands:

   ```
   # /usr/bin/chown oracle:dba /dev/async
   # /usr/bin/chmod 660 /dev/async
   ```

Verifying Asynchronous I/O

To verify asynchronous I/O, first verify that the HP-UX asynchronous driver is configured for Oracle Database, then verify that Oracle Database is executing asynchronous I/O through the HP-UX device driver.

Verifying that HP-UX Asynchronous Driver is Configured for Oracle Database

To verify that the HP-UX asynchronous driver is configured properly for Oracle Database:

1. Start Oracle Database with a few parallel query slave processes.
2. Start the GlancePlus/UX utility, as follows:

   ```
   $ gpm
   ```

3. In the main window, choose Reports>Process List.
4. In the Process List window, select one parallel query slave process and choose Reports>Process Open Files.

   The list of files currently opened by the parallel query slave process appears.
5. In the list of open files, check for the `/dev/async` file or the 101 0x000000 mode.

   If either is in the list, the `/dev/async` file has been opened by the parallel query slave process, and the HP-UX asynchronous device driver is configured properly to enable Oracle processes to execute asynchronous I/O. Make a note of the file descriptor number for the `/dev/async` file.

Verifying that Oracle Database is Using Asynchronous I/O

To verify that Oracle Database is using asynchronous I/O through the HP-UX asynchronous device driver:

1. Attach the HP-UX `tusc` utility to the same Oracle parallel query slave that you chose in GlancePlus in the preceding procedure.
2. Run an I/O bound query in your environment.
3. Check the pattern of read and write calls in the `tusc` output.

   For example, enter the following command, where *`pid`* is the process ID of a parallel query slave supposed to execute asynchronous I/O:

   ```
   $ tusc -p pid > tusc.output
   ```

4. After running the query, press Ctrl+c to disconnect from the process, then open the `tusc.output` file.

   The following example shows a sample `tusc.output` file:

```
( Attached to process 2052: "ora_p000_tpch" [ 64-bit ])
...................
.........................
[2052] read(9, "80\0\001\013  \b\0\0\0\0\0\0\0\0".., 388) .. = 28
[2052] write(9, "\0\0\00e\0\0\0\080\0\001\013Ð \0".., 48) .. = 48
[2052] read(9, "80\0\001\013¢ 18\0\0\0\0\0\0\0\0".., 388) .. = 28
[2052] write(9, "\0\0\00e\0\0\0\080\0\001\01bd4\0".., 48) .. = 48
```

   If the DISK_ASYNCH_IO initialization parameter is not explicitly set to FALSE (set to TRUE by default), the `tusc.output` file shows a pattern of asynchronous read/write calls of the same file descriptor (9 in the preceding example) back-to-back.

   Map the file descriptor number in the `tusc.output` file to that used by `/dev/async` file in GlancePlus. They should match for the particular parallel query slave process. This verifies that I/O through the HP-UX asynchronous driver is asynchronous. With synchronous I/O, or if the DISK_ASYNC_IO initialization parameter is explicitly set to FALSE, you do not see the asynchronous read/write pattern described previously. Instead, you see calls to lseek or pread/pwrite. You also see lots of different file descriptors (the first argument to read/write) instead of just a single file descriptor.

Asynchronous Flag in SGA

Oracle Database on HP-UX uses a non-blocking polling facility provided by the HP-UX asynchronous driver to check the status of I/O operations. This polling is performed by checking a flag that is updated by the asynchronous driver based on the status of the I/O operations submitted. HP-UX requires that this flag be in shared memory.

Oracle Database configures an asynchronous flag in the SGA for each Oracle process. Oracle Database on HP-UX has a true asynchronous I/O mechanism where I/O requests can be issued even though some previously issued I/O operations are not complete. This helps to enhance performance and ensures good scalability of parallel I/O processes.

Before Oracle8*i* release 8.1.7, Oracle Database was able to execute I/O operations only from shared memory using the HP-UX asynchronous driver. Oracle Database 10*g* executes I/O operations from both shared memory and process-private regions

using the new HP-UX asynchronous driver. However, I/O operations through the asynchronous driver are not asynchronous in nature. This is because Oracle Database must perform a blocking wait to check the status of I/O operations submitted to the asynchronous driver. This causes some Oracle processes, for example the database writer process, to essentially execute synchronous I/O.

Large Memory Allocations and Oracle Database Tuning

Applications running on Oracle Database 10*g* can use significantly more memory than applications running on earlier Oracle releases. There are two reasons for this:

- The CURSOR_SPACE_FOR_TIME initialization parameter is changed from the default value FALSE, to TRUE.
- Oracle Database 10*g* changes the default setting for virtual memory data pages from D (4KB) to L (1 GB) on HP-UX systems.

Persistent Private SQL Areas and Memory

When a user submits a SQL statement, Oracle Database automatically performs the following memory allocation steps:

1. It checks the shared pool in the Oracle SGA to see if a shared SQL area for an identical statement already exists. If a shared SQL area exists, Oracle uses it to run subsequent new instances of the statement. If a shared SQL area does not exist, Oracle allocates a new shared SQL area in the shared pool for the SQL statement.
2. Oracle allocates a private SQL area on behalf of the user session.

Private SQL areas contain data such as bind information and runtime memory structures for processed SQL statements. Private SQL areas also contain parsed statements and other statement processing information.

Every user who submits the same SQL statement has a cursor that uses a single shared SQL area. In this way, many private SQL areas can be associated with the same shared SQL area. If a user session is connected through a dedicated server, then private SQL areas are located in the server process PGA. However, if a session is connected through a shared server, part of the private SQL area is kept in the SGA.

The CURSOR_SPACE_FOR_TIME initialization parameter specifies whether a SQL cursor can be deallocated from the library cache to make room for a new SQL statement. When this parameter is set to TRUE, Oracle Database can deallocate a

shared SQL area from an Oracle library cache only when all application cursors associated with the SQL statement are closed. Setting the parameter to TRUE also prevents the deallocation of private SQL areas associated with open cursors, making the user's private SQL area persistent.

Compared to earlier Oracle releases, setting the CURSOR_SPACE_FOR_TIME initialization parameter to TRUE in Oracle Database has the following advantages:

- It accelerates SQL execution calls, because each active cursor's SQL area is present in memory and never aged out.
- It improves application performance, because Oracle Database does not need to verify that a shared SQL area is in the library cache. By retaining private SQL areas between SQL statement executions, it also helps to save cursor allocation and initialization time.

Compared to earlier Oracle releases, setting the CURSOR_SPACE_FOR_TIME initialization parameter to TRUE in Oracle Database has the following disadvantages:

- It increases the memory requirements of user processes because of an increased memory allocation for the persistent private SQL areas.
- It significantly increases cursor memory, which leads to larger memory allocations for Oracle Database shadow processes.

If you set the value of CURSOR_SPACE_FOR_TIME parameter to FALSE, you might experience degraded overall SQL execution and performance. Setting the parameter to FALSE can result in rapid deallocation of shared SQL areas from the library cache.

Default Large Virtual Memory Page Size

By default, Oracle Database uses the largest virtual memory page size setting available on HP-UX for allocating process-private memory. It is defined by the value L, largest, and is currently 1 GB on HP-UX 11i. This value is set as one of the LARGE_PAGE_FLAGS options when linking an Oracle executable.

When the virtual memory page size is set to L, HP-UX allocates the available process-private memory to pages of 1 MB, 4 MB, 16 MB and so on, until it reaches the 1 GB limit, or until it reaches the total amount of allocated memory. If you allocate enough memory to the Oracle PGA for the operating system to be able to allocate memory in larger data page size units, then the operating system allocates the maximum page size at once. For example, if you allocate 48 MB for the Oracle PGA, then your system can have either three pages each of 16 MB, or a series of

pages in unit sizes with the smaller multiples, for example, four 1 MB pages, three 4 MB pages, and two 16 MB pages. If you allocate 64 MB to the PGA, then the operating system allocates one page of 64 MB, as the data page unit size matches the available memory.

In general, large memory pages yield better application performance by reducing the number of virtual memory translation faults that must be handled by the operating system, freeing more CPU resources for the application. Large pages help to reduce the total number of data pages needed to allocate the process-private memory. This reduction decreases the chances of translation lookaside buffer (TLB) misses at the processor level.

However, if applications are constrained in memory and tend to run a very large number of processes, then this drastic page size increase might lead processes to indicate large memory allocations, followed by an out of memory error. If this happens, you must lower the page size to a value between the D (default) size of 4 KB and the L (largest) size of 1 GB.

With the lowest page size setting (4 KB), CPU utilization can be 20% higher than that with a larger page size setting. With the highest setting of L, the memory utilization can be 50% higher than that with a 4 MB setting. In cases where the system shows memory constraints, Oracle recommends that you set the page size to match the requirements of the particular application, within the constraints of available memory resources.

For example, an application that has problems with the L setting might show reasonable performance with a 4 MB virtual memory page setting.

Tuning Recommendations

To address tuning for the increased memory allocation required for persistent private SQL areas and large virtual memory page sizes, Oracle recommends the following:

- Keep the value of the CURSOR_SPACE_FOR_TIME parameter to TRUE, unless the system indicates library cache misses when running the application. If that happens, the shared pool might be small enough to hold the SQL areas for all concurrent open cursors.
- Decrease the virtual memory data page size for Oracle Database as necessary. Use the following command to alter the page size setting:

  ```
  # /usr/bin/chatr +pd newsize $ORACLE_HOME/bin/oracle
  ```

In the preceding example, *newsize* represents the new value of the virtual memory page size.

Display the new setting using the `chatr` command, as follows:

```
# /usr/bin/chatr $ORACLE_HOME/bin/oracle
```

Oracle Real Application Clusters on HP-UX

The following section provides information about Oracle Real Application Clusters on HP-UX.

PA-RISC Only: Tuning Hyper Messaging Protocol Parameters

You can use Hyper Messaging Protocol (HMP) as the cluster interconnect protocol for Oracle Real Application Clusters on HP-UX 11i (11.11) PA-RISC systems. For more information about HMP, see the HP documentation.

To enable Oracle to use HMP, you must relink the `oracle` binary with HMP, as follows:

1. Set up the environment on all nodes.
2. Shut down the database.
3. Enter the following commands on all nodes:

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk ipc_hms ioracle
```

If you choose to use HMP, Oracle recommends that you review and tune the HMP parameters listed in the `/opt/clic/lib/skgxp/skclic.conf` file.

CPU_COUNT Initialization Parameter and HP-UX Dynamic Processor Reconfiguration

HP-UX 11i supports dynamic runtime reconfiguration of processor sets, Psets, and dynamic reassignment of workload between processor sets by valid users.

On PA-RISC systems, HP-UX Virtual Partitions (vPars) enable users to configure their systems in multiple logical partitions where each partition is assigned its own set of processors, memory, and I/O resources, and can run a separate instance of the HP-UX operating system. HP-UX Processor Sets integrated with vPars support

dynamic processor migration from one virtual partition to another without requiring a reboot of any virtual partition. This helps to provide efficient resource partitioning between applications to minimize interference and guarantees necessary resource allocation to each application running on the HP-UX server.

The Oracle Database CPU_COUNT initialization parameter specifies the number of CPUs available to Oracle Database. Oracle Database 10*g* on HP-UX 11i can dynamically detect changes to the CPU host configuration by periodically querying the operating system. If there are any changes to the number of CPUs in the system, Oracle adjusts the CPU_COUNT parameter to the correct value to reallocate its internal resources. This allows new workloads to take advantage of the newly added processors, and database performance can improve without any changes by the DBA, provided high CPU usage was the cause of the bottleneck.

Some initialization parameter values are calculated based on the CPU_COUNT value at system startup. If changes occur to the number of CPUs after system startup, Oracle does not dynamically update these initialization parameters to account for the new number of CPUs. This might sometimes result in suboptimal database configuration if the new number of CPUs is significantly different from the original. If the number of CPUs on a system increases significantly, the database might not take advantage of on the additional processing power.

If the number of CPUs on the system increases by a small number, for instance from 2 to 4, you do not need to take any action.

If the number of CPUs on the system increases by a large number, for instance from 2 to 32, follow these steps:

1. Use one of the following methods to set the CPU_COUNT initialization parameter to the new value:
   - If the database uses a server parameter file (`spfile`*`dbname`*`.ora`), enter the following SQL*Plus command as the SYS user to specify a new value for the parameter:

     ```
     SQL> ALTER SYSTEM SET CPU_COUNT=32 SCOPE=SPFILE
     ```

   - If the database uses an initialization parameter file (`init`*`sid`*`.ora`), edit the file and specify a new value for the parameter.
2. Restart the database.

C

Administering Oracle Database on Linux

This appendix contains information about administering Oracle Database on Linux. It contains the following sections:

Linux x86 Only: Extended Buffer Cache Support

Oracle Database can allocate and use more than 4 GB of memory for the database buffer cache. This section describes the limitations and requirements of the extended buffer cache feature on Linux x86.

> **See Also:** For more information about the extended buffer cache feature, see *Oracle Database Concepts*.

In-Memory File System

To use the extended buffer cache feature, create an in-memory file system on the `/dev/shm` mount point equal in size or larger than the amount of memory that you intend to use for the database buffer cache. For example, to create an 8 GB file system on the `/dev/shm` mount point, follow these steps:

1. Enter the following command as the `root` user:

   ```
   # mount -t shm shmfs -o size=8g /dev/shm
   ```

2. Add an entry in the `/etc/fstab` file, similar to the following, to ensure that the in-memory file system is mounted when the system reboots:

   ```
   shmfs /dev/shm shm size=8g 0 0
   ```

When Oracle Database starts with the extended buffer cache feature enabled, it creates a file in the `/dev/shm` directory that corresponds to the Oracle buffer cache.

Note: If an in-memory file system is already mounted on the `/dev/shm` mount point, ensure that its size is equal to or larger than the amount of memory that is used for the database buffer cache.

USE_INDIRECT_DATA_BUFFERS Initialization Parameter

To enable the extended buffer cache feature, set the USE_INDIRECT_DATA_BUFFERS initialization parameter to TRUE in the parameter file. Doing this allows Oracle Database to specify a larger buffer cache.

Dynamic Cache Parameters

If the extended cache feature is enabled, you must use the DB_BLOCK_BUFFERS parameter to specify the database cache size.

Do not use the following dynamic cache parameters while the extended buffer cache feature is enabled:

- DB_CACHE_SIZE
- DB_2K_CACHE_SIZE
- DB_4K_CACHE_SIZE
- DB_8K_CACHE_SIZE
- DB_16K_CACHE_SIZE

Limitations

The following limitations apply to the extended buffer cache feature:

- You cannot create or use tablespaces with non-default block sizes. You can create tablespaces using only the block size specified by the DB_BLOCK_SIZE parameter.
- You cannot change the size of the buffer cache while the instance is running.

See Also: For more information about the default block size used by the CREATE TABLESPACE command, see the *Oracle Database SQL Reference*.

Note: The default VLM window size is 512 MB. This memory size is allocated to the process's address space. To increase or decrease this value, set the VLM_WINDOW_SIZE environment variable to the new size in bytes. For example, to set the VLM_WINDOW_SIZE to 256 MB, enter the following:

```
$ export VLM_WINDOW_SIZE=268435456
```

The value that you specify for the VLM_WINDOW_SIZE environment variable must be a multiple of 64 KB.

Red Hat Enterprise Linux 3 Only: VLM Window Size

To accommodate the VLM window size, you must increase the default maximum size of the per-process locked memory. To increase it, add the following lines to the `/etc/security/limits.conf` file, where `oracle` is the user that administers the database:

```
oracle          soft    memlock         3145728
oracle          hard    memlock         3145728
```

If you use `ssh` to log in to the system, add the following line to the `/etc/ssh/sshd_config` file to enable the default values to be used when an `ssh` session is started:

```
UsePrivilegeSeparation no
```

Linux x86 Only: Enabling Large Pages on Red Hat Enterprise Linux AS 2.1 and SuSE Linux Enterprise Server 8

Enabling large pages can improve scalability by reducing virtual memory overhead in the kernel. However, memory resources in the large page pool cannot be swapped, so you must be sure that your system has sufficient memory.

Note: Using the bigpages parameter is supported only on Red Hat Enterprise Linux AS 2.1 and SuSE Linux Enterprise Server 8. It is not available or supported on Red Hat Enterprise Linux AS 3.

To use the large pages feature, you must determine the bigpages parameter value. To determine this value, convert the size of the instance's SGA to megabytes, and round up by 4 MB. Use the following example as a guide:

Note: You can determine the size of the SGA by using the SHOW SGA command in SQL*Plus or by using the `ipcs` command.

If the SGA is 2.7 GB:

1. Determine the bigpages value as follows:

   bigpages = 2.7 * 1024 = 2764.8

2. Round the value up by 4 to 2768.

3. Depending on your boot loader, perform one of the following steps:

   LILO:

   a. Add the `bigpages` option to the appropriate image section in the `/etc/lilo.conf` file:

      ```
      append = "bigpages=2768MB"
      ```

   b. Run `/sbin/lilo`.

   c. Restart the system.

   GRUB:

   a. Add the `bigpages` option to the kernel command in the `/etc/grub.conf` file, for example:

      ```
      kernel /vmlinuz-2.4.9 root=/dev/hda5 bigpages=2768MB
      ```

   b. Restart the system.

4. To enable the large pages feature, set the value of the shm-use-bigpages kernel parameter in the `/etc/sysctl.conf` file to either 1 or 2, depending on whether you are using the extended buffer cache feature, and set the USE_INDIRECT_DATA_BUFFERS initialization parameter to TRUE in the parameter file:

   - If you are not using the extended buffer cache (VLM), set the value to 1:

     ```
     kernel.shm-use-bigpages = 1
     ```

   - If you are using the extended buffer cache (VLM), set the value to 2 to configure the bigpages memory pool to include the memory allocated in `/dev/shm`:

     ```
     kernel.shm-use-bigpages = 2
     ```

5. Enter the following command to set the parameter values:

```
# sysctl -p /etc/sysctl.conf
```

6. Start up Oracle Database.

Using hugetlbfs on Red Hat Enterprise Linux AS 2.1 (Linux Itanium)

To enable Oracle Database on Red Hat Enterprise Linux AS 2.1 to use TLB memory, set the value of the `vm.nr_hugepages` kernel parameter. To determine the parameter value required, use the following formula to calculate the number of huge TLB pages that you need to reserve:

(total SGA size for instance + hugeTLB page size - 1)/(huge TLB page size)

Ensure that the value of the shmmax kernel parameter is at least the size of a huge TLB page.

For example, if `/proc/meminfo` lists the huge TLB page size as 256 MB, and the total SGA size for the instance is 3.9 GB, then set the value for `vm.nr_hugepages` to (3.9 GB + 256 MB - 1)/256 MB = 16.

Using hugetlbfs on Red Hat Enterprise Linux AS 3

To use hugetlbfs on Red Hat Enterprise Linux AS 3, perform the following:

1. Determine the memory required for the large page pool, similar to the large pages example in the previous section (2768 MB).
2. Depending on your boot loader, perform one of the following:

   LILO:

   a. Add the `hugepages` option to the appropriate image section in the `/etc/lilo.conf` file, specifying the number of pages:

      ```
      append = "hugepages=1024"
      ```

   b. Run `/sbin/lilo`.

   c. Restart the system.

GRUB:

a. Add the `hugepages` option to the `kernel` command in the `/etc/grub.conf` file, specifying the number of pages:

```
kernel /vmlinuz-2.4.9 root=/dev/hda5 hugepages=1024
```

b. Restart the system.

3. Add or edit the following entry in the `/etc/sysctl.conf` file, specifying the large page pool size in megabytes:

```
vm.hugetlb_pool = 2768
```

4. Enter the following command to set the kernel parameter values:

```
# sysctl -p /etc/sysctl.conf
```

5. Verify that this amount of memory was moved successfully into the hugetlb pool, as follows:

```
# cat /proc/meminfo
```

The lines at the end of the display show how many memory pages were moved into the hugetlb pool.

6. Start up the database.

Linux x86 Only: Increasing SGA Address Space

Depending on your distribution of Linux, use the instructions in one of the following sections to increase the SGA address space:

- Red Hat Enterprise Linux AS 2.1 and SuSE Linux Enterprise Server 8
- Red Hat Enterprise Linux AS 3

Red Hat Enterprise Linux AS 2.1 and SuSE Linux Enterprise Server 8

To increase the SGA address space on Red Hat Enterprise Linux AS 2.1, complete the following:

1. If necessary, log in as the `oracle` user.
2. In the `$ORACLE_HOME/rdbms/lib` directory, enter the following commands:

```
$ genksms -s 0x15000000 > ksms.s
$ make -f ins_rdbms.mk ksms.o
$ make -f ins_rdbms.mk ioracle
```

> **Note:** If Oracle Database does not start after completing this procedure, or if there are runtime memory errors, then increase the hexadecimal number specified in the first command. For example, if the 0x15000000 value prevents Oracle Database from starting, specify the value 0x20000000. Lowering this value increases the SGA address space, but could decrease the PGA address space.

3. Enter the following command to determine the process ID of the `oracle` user's shell process:

```
$ echo $$
```

   The number returned is the process ID.

4. Switch user to `root`:

```
$ su - root
```

5. Enter the following commands to change the mapped base setting for the `oracle` user's shell process, where *`pid`* is the process ID identified in step 3:

```
# echo 268435456 > /proc/pid/mapped_base
```

6. Enter `exit` to return to the `oracle` user's shell process, and start the Oracle Listener and Oracle Database.

> **Note:** All Oracle processes need to get this modified mapped base value. Starting the listener from the shell that has the modified mapped base allows client connections to connect properly.

Red Hat Enterprise Linux AS 3

To increase the SGA address space on Red Hat Enterprise Linux AS 3, follow these steps:

1. If necessary, log in as the `oracle` user.
2. In the `$ORACLE_HOME/rdbms/lib` directory, enter the following commands:

```
$ genksms -s 0x15000000 > ksms.s
$ make -f ins_rdbms.mk ksms.o
$ make -f ins_rdbms.mk ioracle
```

3. Start Oracle Database.

Asynchronous I/O Support

> **Note:** On Linux, Automatic Storage Manager (ASM) uses asynchronous I/O by default.

Oracle Database supports kernel asynchronous I/O. This feature is disabled by default. If you are running Oracle Database on a system that supports kernel asynchronous I/O and is certified by Oracle to use asynchronous I/O, perform the following steps to enable asynchronous I/O support:

1. As the `oracle` user, change directory to the `$ORACLE_HOME/rdbms/lib` directory.
2. Enter the following command:

```
$ make -f ins_rdbms.mk async_on
```

> **Note:** If you receive the `"/usr/bin/ld: cannot find -laio"` error, then the system does not support kernel asynchronous I/O and you must enter the following command to restore the Oracle instance to a usable state:
>
> ```
> $ make -f ins_rdbms.mk async_off
> ```

By default, the DISK_ASYNCH_IO initialization parameter in the parameter file is set to TRUE to enable asynchronous I/O on raw devices. To enable asynchronous I/O on file system files:

1. Ensure that all Oracle database files are located on file systems that support asynchronous I/O.
2. Set the FILESYSTEMIO_OPTIONS initialization parameter in the parameter file to ASYNCH.

Direct I/O Support

Direct I/O support is not available and is not supported on Red Hat Enterprise Linux 2.1 and SuSE Linux Enterprise Server 8. It is available and is supported on Red Hat Enterprise Linux 3 if the driver being used on the system supports varyio. To enable direct I/O support:

- Set the FILESYSTEMIO_OPTIONS initialization parameter to DIRECTIO.
- If you are using the asynchronous I/O option, set the FILESYSTEMIO_OPTIONS initialization parameter to SETALL.

Semtimedop Support

Oracle Database 10*g* supports the `semtimedop()` system call (semaphores with a time limitation). This feature is disabled by default. If you are running Oracle Database on SuSE Linux Enterprise Server 8 or Red Hat Enterprise Linux AS 3, you can enter the following command as the `oracle` user in the `$ORACLE_HOME/rdbms/lib` directory to enable support for the feature:

```
$ make -f ins_rdbms.mk smt_on
```

To disable `semtimedop()` support, enter the following command as the `oracle` user from the `$ORACLE_HOME/rdbms/lib` directory:

```
$ make -f ins_rdbms.mk smt_off
```

Linux x86 Only: High Speed Network Support

On x86 systems only, Oracle Net supports Sockets Direct protocol (SDP) over the InfiniBand network architecture on Red Hat Enterprise Linux AS 2.1 and 3 for Oracle Database 10*g* Release 1. For this release, SDP support is limited to

synchronous I/O only. For information about support for using asynchronous I/O on SDP, check the following Web site for updates:

```
http://otn.oracle.com/products/oraclenet/files/Oracle_Net_High-Speed_Interconnect_Support.doc
```

Note: Do not set the Oracle Net NET_ASYNC_IO and SDP_ASYNC_IO configuration parameters unless otherwise stated on this Web site.

D

Administering Oracle Database on Tru64 UNIX

This appendix contains information about administering Oracle Database on Tru64 UNIX. It contains the following sections:

Enabling Oracle Database Directed Placement Optimizations

HP AlphaServer GS series, ES47 and ES80 systems consist of smaller building blocks called Resource Affinity Domains (RADs). A RAD is a collection of tightly coupled CPUs, memory modules, and an I/O controller coupled through a fast interconnect. A second-level interconnect connects each of the RADs together to form a larger configuration.

Unlike previous generation servers which have only one common shared interconnect between CPUs, memory, and I/O controller, GS series, ES47 and ES80 servers can offer superior performance and memory access times when a particular CPU accesses memory within its own RAD or uses its local I/O controller. Because of the switched interconnect, all I/O activity and memory accesses within one RAD do not interfere with those within another RAD. However, because memory accesses between a CPU and memory module located across RAD boundaries must traverse two levels of interconnect hierarchy, these memory references take longer relative to memory references that are within a RAD.

Directed memory and process placement support allows sophisticated applications to communicate their specific needs for process and memory layout to the operating system. This communication results in greater performance through increased localization of memory references within a RAD.

Oracle Database includes enhanced support for the special capabilities of high performance servers such as the GS series, ES47 and ES80. Directed placement optimizations specifically take advantage of hierarchical interconnects available in these servers. All previous generation servers have a single shared interconnect, so these servers neither directly benefit from directed placement optimizations nor is there any loss of performance on these servers. Therefore, by default, these optimizations are disabled in Oracle Database.

Requirements to Run the Directed Placement Optimizations

The system must meet the following requirements for Oracle Database directed placement optimizations to work:

- The system must be an HP GS series, ES47, or ES80 AlphaServer or similar locality sensitive system. The Oracle Database optimizations only affect systems that are locality sensitive.
- The operating system must be Tru64 UNIX V5.1B or higher.

Enabling Oracle Directed Placement Optimizations

To enable Oracle directed placement optimizations, follow these steps:

1. Shut down the Oracle instance.
2. Relink Oracle Database by entering the following commands:

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk numa_on
$ make -f ins_rdbms.mk ioracle
```

If you are not using a compatible version of Tru64 UNIX, the following message is displayed:

```
Operating System Version Does not Support NUMA.
Disabling NUMA!
```

If you enable Oracle directed placement optimizations, and later change Tru64 UNIX to an incompatible version, disable Oracle directed placement optimizations as described in the following section.

Disabling Oracle Directed Placement Optimizations

To disable Oracle directed placement optimizations, follow these steps:

1. Shut down the Oracle instance.
2. Relink Oracle Database using the `numa_off` option:

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk numa_off
$ make -f ins_rdbms.mk ioracle
```

Using Oracle Directed Placement Optimizations

The Oracle directed placement optimizations assume an equi-partitioned configuration. This means that all RADs are configured with the same number of CPUs and the same amount of memory. Oracle Database is assumed to run across all RADs on the system.

Oracle Initialization Parameters

To make the most efficient use of the local environment, Oracle Database adjusts some initialization parameters automatically depending on the database

configuration as reported by the operating system. This practice eliminates common errors in correctly computing subtle dependencies in these parameters.

Tru64 UNIX Subsystem Attributes

You must set the subsystem attributes in the following table to realize the full benefits of a NUMA system:

Subsystem	Attribute	Setting
ipc	ssm_threshold	0
	shm_allocate_striped	1 (default)
vm	rad_gh_regions[0], rad_gh_regions[1], and so on	Set the rad_gh_regions[*n*] attributes as follows: **1.** Enter the following command: `$ ipcs -b` **2.** From the output, add the values of the SEGSZ column, and convert the total to megabytes, by dividing by 1048576. **3.** If the value of the LOG_BUFFER initialization parameter is less than 8388608 (8 MB), subtract 8388608 from the total. If the LOG_BUFFER value is between 8388608 and 16777216 (16 MB), subtract 16777216 from the total of the SEGSZ output. **4.** Divide the result of step 3 by the number of RADs on the system. **5.** Set the rad_gh_regions[0] parameter to the result of step 4, plus either 8 MB or 16 MB, depending on the value of the LOG_BUFFER initialization parameter and other valid rad_gh_regions[*n*] attributes. These steps assume that all of the RADs on the system are allocated to Oracle. If Oracle is restricted to using a subset of the number of RADs on the system, refer to the "Restricting Oracle Database to a Subset of the Number of RADs on the System" section on page D-5. Instead of the preceding steps, the rad_gh_regions[0] setting must be applied to the rad_gh_regions[*n*] parameter for the first RAD listed in the `numa_config_sid.ora` file.

There are 63 rad_gh_regions attributes in the vm subsystem in Tru64 UNIX V5.1B. Set only the attributes for the total number of RADs on the system. For example, if there are 4 RADs on the system all of which are used by Oracle and the SEGSZ column total is 10248 MB with the LOG_BUFFER initialization parameter set to

2097152 (2 MB), set rad_gh_regions[0] to 2568 and rad_gh_regions[1], rad_gh_regions[2], and rad_gh_regions[3] to 2560. To successfully start the instance, you might have to raise this value slightly, by 1 or 2.

If CPUs and memory are taken off-line, Oracle Database continues to function, but loses performance. If you anticipate frequent off-lining of RADs or equi-partitioning is not feasible, Oracle recommends running Oracle Real Application Clusters, using one instance per RAD. Using Oracle Real Application Clusters, you can configure individual instances with different sets of initialization parameters to match the actual RAD configuration. You can also start up or shut down specific instances without affecting overall application availability.

Process Affinity to RADs

You can improve performance by directing the operating system to run the processes on specific RADs. If connections to the database are made through the Oracle Listener process, and there is a corresponding network interconnect adapter on the RAD, you can run a listener on each RAD. To run the listener on a particular RAD, enter the following command, where *rad_number* is the number of the RAD:

```
$ runon -r rad_number lsnrctl start [listener_name]
```

All Oracle shadow processes are automatically created on the same RAD as the Oracle listener.

Restricting Oracle Database to a Subset of the Number of RADs on the System

You can restrict Oracle Database to run on a subset of the number of RADs on the system. When an instance starts, Oracle Database looks for the `numa_config_`*sid*`.ora` file in the `$ORACLE_HOME/dbs` directory. If the file does not exist, Oracle Database uses all of the RADs on the system. If the file exists, it instructs the instance to use only the RADs specified in the file, and to run processes on only the RADs specified.

The `numa_config_`*sid*`.ora` file has the following format:

```
number_RADs
group1
...
groupn
```

In this example, *number_RADs* is the number of processor groups or RADs that Oracle should use and *group1* to *groupn* are the numbers of the RADs for Oracle to use.

Use the following guidelines when creating the file:

- Do not include any blank lines
- Include each number on a separate line
- Do not include anything else on the line

The RADs are numbered from zero to the maximum for the particular system. If the system is a fully integrated AlphaServer GS320, the RADs are numbered 0,1,2,3,4,5,6,7. For example, if you want to restrict Oracle Database to RADs 1 and 3 of a fully configured GS320, create a `numa_config_sid.ora` file similar to the following:

```
2
1
3
```

The group numbers do not have to be consecutive or in increasing numerical order.

Parallel query slaves and the shadow processes for bequeath connections are created on the RADs specified in the file. For remote TCP connections, you must bind the listener to the RADs explicitly, as described in the "Process Affinity to RADs" section on page D-5.

Supporting Mixed CPU Systems

Tru64 UNIX systems using Tru64 UNIX V5.1B or higher can have mixed CPU speeds and types. All CPUs in a single RAD must have the same speed and cache size. Another RAD can have a set of CPUs with a different speed and cache size.

The performance of a mixed CPU system depends on the proportion of slower CPUs to faster CPUs. Also, performance is affected by the placement of Oracle processes on the system. In a high transaction Online Transaction Processing (OLTP) environment, placing the database writer and log writer processes on the slower CPUs can adversely affect performance. In a data warehousing or decision support environment, placing the database writer and log writer processes on the slower CPUs might not be noticeable at all.

The ability to mix CPU systems enables you to protect your hardware investment. You can add faster and more powerful CPUs to a system without needing to replace older CPUs. HP and Oracle have tested and support mixed CPU systems.

Note: Do not expect the mixed CPU system to perform as well as a system made up entirely of the fastest CPUs of the mixed CPU system. However, a mixed CPU system should perform better than a system made up entirely of the slowest CPUs of the mixed CPU system. Contact HP for a complete list of rules and restrictions for mixed CPU systems.

Gathering Database Statistics on Tru64 UNIX

Oracle Database 10*g* runs only on Tru64 UNIX V5.1B or higher. This is because HP changed the size of the long double data type from 64 bits on Tru64 UNIX V4.0x to 128 bits on Tru64 UNIX V5.*x*. This change causes certain Oracle operations to perform with increased precision. One of these operations stores statistics in the data dictionary after a table or index is analyzed.

The query optimizer within Oracle Database uses the statistics stored in the data dictionary to determine how best to execute a query. If a stored statistic does not match a statistic calculated by the query optimizer while it searches for the best plan, the query optimizer might use the wrong plan to execute the query. This can cause the query to perform poorly or fail.

For this reason, after upgrading from Oracle8*i* release 8.1.7 or lower to Oracle Database 10*g* you should analyze all object statistics for each schema. There is no need to reanalyze any schemas after upgrading from Oracle9*i* release 1 (9.0.1) or release 2 (9.0.2) to Oracle Database 10*g*. You can use the DBMS_STATS.GATHER_SCHEMA_STATS procedure to perform the analysis to gather statistics for each schema. The DBMS_STATS package saves the current table or index statistics in a table in case the new statistics cause problems.

See Also: For more information about gathering database statistics, see the *PL/SQL Packages and Types Reference*.

Oracle Real Application Clusters on Tru64 UNIX

This section describes Oracle Real Application Clusters on Tru64 UNIX.

Reliable Data Gram

Reliable Data Gram (RDG) is an IPC infrastructure for the Tru64 UNIX TruCluster platform. It is the default IPC method for Oracle Database on Tru64 UNIX and is optimized for Oracle Real Application Clusters environments.

Requirements

RDG requires that the node be a member of the cluster and connected through the memory channel. Oracle recommends that you set the node-wide subsystem attributes listed in Table D–1 when using RDG.

Table D–1 rdg Subsystem Attribute Settings

Attribute	Setting
max_objs	At least 5 times the number of Oracle processes per node and up to the larger of 10240 or the number of Oracle processes multiplied by 70.
msg_size	Equal to or greater than the maximum value of the DB_BLOCK_SIZE parameter for the database. Oracle recommends a value of 32768 because Oracle Database supports different block sizes for each tablespace.
max_async_req	At least 100 or the operating system default, whichever is larger. **Note**: A value of 1000 or greater might provide better performance.
max_sessions	170 (20 + value of Oracle PROCESSES initialization parameter).
rdg_max_auto_msg_wires	Must be set to 0.

Enabling UDP IPC

With Oracle Database 10*g*, RDG is the default IPC method on Tru64 UNIX. When the Oracle Real Application Clusters option is enabled, the Global Cache Service (GCS), Global Enqueue Service (GES), Interprocessor Parallel Query (IPQ), and Cache Fusion use RDG. The User Datagram Protocol (UDP) IPC implementation is still available but you must enable it explicitly.

You must enable the Oracle Real Application Clusters option before enabling UDP IPC. To enable the Oracle Real Application Clusters option, use the Oracle Universal Installer or enter the following commands:

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk rac_on
$ make -f ins_rdbms.mk ioracle
```

To make the Oracle IPC routines use the UDP protocol, you must relink the `oracle` executable. Before performing the following steps, shut down all instances in the cluster.

To enable UDP IPC, enter the following commands:

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk ipc_udp
$ make -f ins_rdbms.mk ioracle
```

To disable UDP IPC and revert to the default implementation (RDG) for Oracle Real Application Clusters, enter the following commands:

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk rac_on
$ make -f ins_rdbms.mk ioracle
```

For information about UDP and setting the CLUSTER_INTERCONNECTS parameter, refer to the CLUSTER_INTERCONNECTS Initialization Parameter section on page 1-11.

Simultaneous Enabling of Oracle Real Application Clusters and NUMA

Oracle certifies that Oracle Real Application Clusters and Directed Placement (NUMA) can run simultaneously. For most workloads on AlphaServer GS series, ES47, and ES80 servers, enabling the NUMA option on systems running Oracle Database 10*g* release 1 (10.1) and Real Application Clusters provides significant performance improvements.

Tuning Asynchronous I/O

Oracle Database for Tru64 UNIX systems can perform either synchronous or asynchronous I/O. To improve performance, Oracle recommends that you use asynchronous I/O. Set the DISK_ASYNC_IO initialization parameter to TRUE to enable asynchronous I/O.

Oracle Database can use asynchronous I/O on any datafiles that are stored on AdvFS or clustered file systems (CFS), in Automatic Storage Management disk groups, or on raw devices. You must tune some kernel subsystem attributes for optimal asynchronous I/O performance.

aio_task_max_num Attribute

Set the aio_task_max_num kernel subsystem attribute for a single instance to 8193.

You should adjust the setting of the aio_task_max_num attribute to accommodate any other applications that use asynchronous I/O, including multiple Oracle Database instances on a single node. Set the value of the parameter to the maximum number of I/O requests that any application can issue. For example, if three applications are running on a system and application 1 can issue a maximum of 10 simultaneous asynchronous I/O requests, application 2 can issue 100 simultaneous asynchronous I/O requests, and application 3 can issue 1000 simultaneous asynchronous I/O requests, set the aio_task_max_num parameter to at least 1000.

If you do not set the aio_task_max_num attribute as described in this section, the performance of Oracle Database is reduced and spurious I/O errors might occur. These errors are recorded in the alert log and trace files.

Direct I/O Support and Concurrent Direct I/O Support

This section describes support for direct and concurrent I/O.

Single Instance Requirements

Oracle Database has the following requirements for single instance installations:

- Tru64 UNIX V5.1B or later with the appropriate patch kits.

  See Also: For more information about Tru64 UNIX patch kits, see the *Oracle Database Installation Guide for UNIX Systems*.

- Oracle data files stored on an AdvFS file system or in an ASM disk group.
- The disks that use the AdvFS file system must be physically connected to the computer running the Oracle Database instance. This includes disks attached by fiber channel. This specifically excludes cases where I/O must be served by another node because of a lack of physical connectivity.

On Tru64 UNIX V5.1B systems and higher in a non-clustered system environment, the AdvFS file system and direct I/O give almost all of the performance of raw devices because the file system cache is not used. In addition to this, the file system allows you to more easily manage the database files.

Clustered Systems

On V5.1B systems and higher, Tru64 UNIX supports Clustered File Systems (CFS). CFS provides a single namespace file system for all nodes in a cluster. All file systems mounted in a cluster are automatically seen by all nodes in the cluster. Because it is layered on top of the AdvFS file system, the CFS file system inherits much of the characteristics of non-clustered systems.

Tru64 UNIX V5.1B Clustered Systems

Oracle supports CFS only on Tru64 UNIX V5.1B or later because this file system now supports a concurrent direct I/O model. Any node that has physical connectivity to a drive can issue data I/O to its file systems without consulting with the owning node.

All metadata changes to a file, for example extending, closing, changing the access or modification date, are still served by the owner node and can still cause cluster interconnect saturation. Therefore, it is possible for the CREATE TABLESPACE, ALTER TABLESPACE, ADD DATAFILE, ALTER DATABASE DATAFILE, or RESIZE commands to perform poorly on a CFS file system when compared to raw devices.

Multiple Instance Requirements (Oracle Real Application Clusters)

If you use the CFS file system for data file storage, Oracle Real Application Clusters requires that Direct I/O is enabled on your system. The disks that use the CFS file system must be physically connected to all computers running the Oracle instances. This includes disks attached by fiber channel. It excludes cases where I/O must be served by another node because of physical connectivity.

If the database is running in archivelog mode and the archive logs are being written to disk, the destination CFS domain should be served by the node of the instance that is archiving the redo log. For example, if you have a three-node cluster with one instance on each node (nodea, nodeb, and nodec), you must also have three archive destination CFS domains (arcnodea, arcnodeb, and arcnodec). The domains should be served by nodea, nodeb, and nodec respectively and the

LOG_ARCHIVE_DEST initialization parameter for each instance should specify their respective locations.

Disabling Direct I/O Support

Oracle Database running on an AdvFS file system with direct I/O support enabled should perform as well as Oracle Database running on raw devices. In most cases, an Oracle Database that is stored on AdvFS volumes with direct I/O support enabled should perform the same as or better than the same database with direct I/O support disabled. However, the following workload attributes can reduce performance when direct I/O support is enabled:

- A high read to write ratio
- Oracle data blocks not cached in the SGA because the query uses parallel query slaves
- A UNIX buffer cache (UBC) several megabytes or larger
- Full table scan queries where the same set of tables are scanned repeatedly
- Tables being scanned can fit in the UBC

When direct I/O support is disabled, workloads that have most of the attributes in the preceding list rely heavily on the UBC. Because most if not all of the tables being scanned are cached in the UBC, the I/O requests issued by the parallel query are met by the UBC. As a result, the query completes much faster than if all of the data had to be read from disk, as it would with direct I/O enabled.

When direct I/O support is enabled, Oracle data blocks are not cached in the UBC. They are read into process-private memory instead. This means that any query that reads a previously-scanned table must perform I/O requests to disk to retrieve the data. Disk I/O latencies are several orders of magnitude slower than memory latencies. Therefore, the query runs slower and performance suffers.

If your workload has most of the attributes described in the preceding list, disabling direct I/O support will probably improve performance. However, often there are many different types of queries running on the system at the same time. Some queries only read data while others insert, modify, or delete data and the ratio of the various types of queries differ from site to site. Generally, if your site has more of an OLTP workload, disabling direct I/O support does not improve performance.

Direct I/O support is enabled by default with Oracle Database 10*g*. The undocumented _TRU64_DIRECTIO_DISABLED initialization parameter that is used to disable direct I/O support in Oracle9*i* release 1 (9.0.1) is removed in Oracle Database 10*g*. The generic FILESYSTEMIO_OPTIONS initialization parameter is

used instead. The following table describes the valid values for the FILESYSTEMIO_OPTIONS initialization parameter as interpreted on Tru64 UNIX:

Value	Description
DIRECTIO	Implies that direct I/O support is enabled but asynchronous I/O support is not enabled for I/O to files on an AdvFS files system.
ASYNCH	Equivalent to `none` because asynchronous I/O support is enabled for AdvFS files only if direct I/O support is also enabled.
SETALL	Implies that both direct I/O and asynchronous I/O support are enabled for AdvFS files. This is the default option.
NONE	Disables both direct I/O support and asynchronous I/O support on AdvFS files.

> **See Also:** See the *Oracle Database Reference* for more information about the FILESYSTEMIO_OPTIONS initialization parameter.

The DISK_ASYNCH_IO initialization parameter controls the asynchronous I/O state for all database files, whether they are on file systems or raw devices. Therefore, if the DISK_ASYNCH_IO initialization parameter is set to FALSE, all I/O requests to file system files are synchronous regardless of the value of the FILESYSTEMIO_OPTIONS initialization parameter. The DISK_ASYNCH_IO initialization parameter defaults to TRUE.

Enabling Access to the Real Time Clock

Many Oracle processes are timed, especially if the TIMED_STATISTICS initialization parameter is set to TRUE. These timing functions call the Tru64 UNIX kernel and can affect Oracle Database performance. On Tru64 UNIX, you can improve performance on heavily loaded systems by enabling processes to directly access the real time clock.

To enable access to the real time clock:

1. Log in as the `root` user.
2. Enter the following commands:

```
# mknod /dev/timedev c 15 0
# chmod a+r /dev/timedev
```

If your system is a cluster running Tru64 UNIX V5.1B or higher, enter these commands on one cluster node.

Note: The special file `/dev/timedev` remains on the system after rebooting.

3. Restart the Oracle Database instance.

   The system checks for the existence of the `/dev/timedev` file only on instance startup.

Oracle recommends that you enable this feature on all instances in a cluster, and therefore on all nodes.

Setting Up Raw Devices

Caution: Do not attempt to set up raw devices without the help of an experienced system administrator and specific knowledge about the system that you are using.

Note: To simplify database file management, Oracle recommends that you use ASM or AdvFS with direct I/O in preference to raw devices.

To set up raw devices/volumes on Tru64 UNIX systems:

1. If you are using Oracle Real Application Clusters, make sure that the partitions you are adding are on a shared disk.
2. Determine the names of the free disk partitions.

   A free disk partition is one that is not used for a Tru64 UNIX file system that complies with the following restrictions:

   - It is not listed when you enter the `/usr/sbin/mount` command.
   - It is not in use as a swap device.
   - It does not overlap a swap partition.

- It is not in use by other Tru64 UNIX applications (for example, other instances of the Oracle Database).
- It does not overlap the Tru64 UNIX file system.
- It does not use space already used by the file system.

To determine whether a partition is free, obtain a complete map of the starting locations and sizes of the partitions on the device and check for free space. Some partitions may contain file systems that are currently not mounted and are not listed in the `/usr/sbin/mount` output.

Note: Make sure that the partition does *not* start at cylinder 0.

---

3. Set up the raw device for use by Oracle Database.

   Begin by verifying that the disk is partitioned. If it is not, use the `disklabel` command to partition it.

4. Enter the `ls` command to view the owner and permissions of the device file. For example:

   ```
   $ ls -la
   ```

5. Make sure that the partition is owned by the Oracle software owner. If necessary, use the `chown` command to change the ownership on the block and character files for the device. For example:

   ```
   # chown oracle:dba /dev/rdisk/dsk10c
   ```

6. Make sure that the partition has the correct permissions. If necessary, use the `chmod` command to make the partition accessible to only the Oracle software owner. For example:

   ```
   # chmod 600 /dev/rdisk/dsk10c
   ```

7. Create a symbolic link to the raw devices you require. For example:

   ```
   $ ln -s /dev/rdisk/dsk10c /oracle_data/datafile.dbf
   ```

   To verify that you have created the symbolic link, use the character special device (not the block special device) and enter the following command:

   ```
   $ ls -Ll datafile
   ```

The following output should appear:

```
crwxrwxrwx oracle dba datafile
```

Caution: This symbolic link must be set up on each node of the cluster. Check that no two symbolic links specify the same raw device.

8. Create or add the new partition to a new database.

   To create a new partition, from SQL*Plus enter the following command:

   Note: The size of an Oracle datafile created in a raw partition must be at least 64 KB plus one Oracle block size smaller than the size of the raw partition.

```
SQL> CREATE DATABASE sid
  2  LOGFILE '/oracle_data/log1.dbf' SIZE 100K
  3 '/oracle_data/log2.dbf' SIZE 100K
  3  DATAFILE '/oracle_data/datafile.dbf' SIZE 10000K REUSE;
```

   To add a partition to a tablespace in an existing Oracle database, enter:

```
SQL> ALTER TABLESPACE tablespace_name
  2  ADD DATAFILE '/dev/rdisk/dsk10c' SIZE 10000K REUSE;
```

You can use the same procedure to set up a raw device for the redo log files.

Spike Optimization Tool

The Spike optimization tool (Spike) is a performance optimization tool that increases the performance of a Tru64 UNIX binary. In a testing environment, Spike, with feedback, increased the performance of Oracle Database by up to 23 percent on an OLTP workload.

For information about Spike, see the Tru64 UNIX documentation or enter one of the following commands:

- `man spike`
- `spike`

Oracle Database requires Spike version V5.2: (510 USG) GEM: 48C5K LIBMLD: 2.4 DATE: Sep 28 2003 or later.

> **Note:** If you have a version of Spike earlier than V5.2: (510 USG) GEM: 48C5K LIBMLD: 2.4 DATE: Sep 28 2003, contact HP for a patch kit.

Enter the following command to check the version of Spike:

```
$ spike -V
```

You can download the latest version of Spike from the HP Web site.

> **Note:** Oracle does not support versions of the Oracle executable optimized using the `spike` command. If you encounter a problem in an Oracle Database binary that has been optimized using Spike, reproduce the problem with the original unoptimized binary. If the problem persists, see the Preface for information about Oracle services and support.

Using Spike

This section describes the system resources required by Spike, how and why to use Spike optimization flags, and the various ways to run Spike.

Setting System Resources

Table D–2 lists the system resources required to run Spike.

Table D–2 System Resource Requirements for Spike

Resource	Minimum Value
Physical memory	1024 MB
max-per-proc-address-space subsystem attribute value	1024 MB
max-per-proc-data-space subsystem attribute value	1024 MB
vm-maxvas subsystem attribute value	1024 MB

To set the value of these subsystem attributes in the `/etc/sysconfigtab` file, add the following lines:

```
proc:
    max-per-proc-address-space = 0x40000000
    max-per-proc-data-size = 0x40000000
vm:
    vm-maxvas = 0x40000000
```

Set the limits in your shell environment to the highest values. For the C shell, enter:

```
% limit datasize unlimited
% limit memoryuse unlimited
% limit vmemoryuse unlimited
```

Spike can run out of virtual memory if the stacksize limit is set too high. To avoid this problem, enter the following C shell command:

```
% limit stacksize 8192
```

Checking Optimization Flags

Spike provides a large number of optimization flags. However, you cannot use all `spike` command optimizations with Oracle Database. The following Spike optimization flags are certified to run with Oracle Database:

```
-arch, -controlOpt, -fb, -feedback, -map, -nosplit, -nochain, -noporder,
-noaggressiveAlign, -o, optThresh, -splitThresh, -symbols_live, -tune, -v, -V
```

When you run Spike, it places a copy of the optimization flags in the image header comment section of the binary that you are optimizing. Oracle Database checks Spike optimizations used on itself at the beginning of instance startup. If Oracle Database detects an optimization not known to work for the Oracle Database binary, or if the binary had been previously optimized with OM (the predecessor to Spike from HP), the instance startup fails with an ORA-4940 error message. If the instance startup fails, check the alert log file for more information.

Note: Oracle Database requires that you use the Spike `-symbols_live` optimization flag.

Running Spike

Use one of the following methods to optimize an executable using Spike:

- Static spiking
- Running Spike with feedback

Static spiking requires only a few set-up steps and yields approximately half the performance benefit possible compared to running Spike with feedback.

Running Spike with feedback includes all of the optimizations of static spiking plus additional optimizations that are workload-related. Running spike with feedback provides the best possible performance benefit, however, it requires considerably more effort than static spiking.

For both running Spike with feedback and static spiking, Oracle recommends running the spiked Oracle binary in a test environment before moving it to a production environment.

Static Spiking

Static spiking performs optimizations that are not specific to your workload, such as manipulating the global pointer (gp) register and taking advantage of the CPU architecture. In a test environment, roughly half of the performance optimization gain possible from Spike was through static spiking. Furthermore, static spiking is relatively straight-forward and simple. The combination of simplicity and performance gain makes static spiking worth the effort.

Perform the following steps to use static spiking:

1. Shut down the database.
2. Spike the `oracle` image by entering the following command:

   ```
   $ spike oracle -o oracle.spike -symbols_live
   ```

3. Save the original image and create a symbolic link to the spiked image by entering the following commands:

   ```
   $ mv oracle oracle.orig
   $ ln -s oracle.spike oracle
   ```

4. Start up the database.

Note: Before contacting Oracle for support, you must use the original image to reproduce any problems.

Running Spike with Feedback

Running Spike with feedback performs all of the same optimizations as static spiking plus optimizations that are workload-related such as hot and cold basic block movement. In a test environment, approximately half of the performance optimizations gained from Spike was due to the optimizations that depend on feedback information. Running Spike with feedback requires multiple steps and considerably more effort than static spiking. However, performance sensitive customers may find the extra effort worthwhile.

Perform the followings steps to run Spike with feedback:

1. Instrument the Oracle binary by entering the following command:

   ```
   $ pixie -output oracle.pixie -dirname dir -pids oracle_image
   ```

   In the preceding example, *oracle_image* is your original image and *dir* is the name of the directory into which the instrumented executable writes the profiling data files.

   > **Note:** The `-dirname` option saves the `oracle.Counts.`*pid* files in the *dir* directory. Because these files are large and may be numerous depending on the workload, make sure that there is enough free disk space.

   This step also creates an `oracle.Addrs` file that is required later.

   The output of the `pixie` command might contain errors. You can safely ignore these errors.

2. Shut down the database.

3. Save the original image and create a symbolic link to the pixie image by entering the following commands:

   ```
   $ mv oracle oracle.orig
   $ ln -s oracle.pixie oracle
   ```

4. Start up the database and run your workload.

   You cannot run as many users as you can with the standard executable because the `pixie` executable is larger and slower. As you use Oracle Database, several `oracle.Counts.`*pid* files are created, where *pid* is the process ID for the corresponding Oracle process. Keep track of the process ID of each Oracle

process for which the optimization is aimed. These could be the shadow Oracle processes of the clients.

5. Shut down the database.
6. Create a symbolic link to replace the original executable by entering the following command:

   ```
   $ ln -s oracle.orig oracle
   ```

7. If you can identify one `oracle.Counts.`*`pid`* file as representative of your workload, perform step a. If you must merge several counts files together to better represent your workload, perform step b.

   a. Make sure that the `oracle.Addrs` file created by the `pixie` command, the `oracle.Counts.`*`pid`* files, and the original Oracle executable are available.

      Use the process ID (`pid`) to pick a representative `oracle.Counts.`*`pid`* file and then copy it by entering the following command:

      ```
      $ cp oracle.Counts.pid oracle.Counts
      ```

   b. Use the `prof` utility to merge several `oracle.Counts.`*`pid`* files. See the `prof` man pages for more information about this utility.

      If you are using the parallel query option, merge the `oracle.Counts.`*`pid`* files generated by the query slaves and the query coordinator, which is the shadow Oracle process of the query-initiating client.

      If you are not using the parallel query option, merge the `oracle.Counts.`*`pid`* files from the Oracle foreground processes that use the most memory.

      To merge the `oracle.Counts.`*`pid`* files, enter the following command:

      ```
      $ prof -pixie -merge oracle.Counts $ORACLE_HOME/bin/oracle \
      oracle.Addrs oracle.Counts.pid1 oracle.Counts.pid2
      ```

8. Make sure that the `oracle.Addrs` and `oracle.Counts` files are available in the current directory, then run Spike using the feedback information by entering the following command:

   ```
   $ spike oracle -fb oracle -o oracle.spike_fb -symbols_live
   ```

   The output of the `spike` command might contain errors. You can safely ignore these errors.

9. Create a symbolic link to the new `oracle` image by entering the following command:

   ```
   $ ln -s oracle.spike_fb oracle
   ```

10. Start up the database.

E

Administering Oracle Database on Solaris

This appendix contains information about administering Oracle Database on Solaris. It contains the following sections:

- Correcting Undefined Symbols
- Intimate Shared Memory

Correcting Undefined Symbols

Oracle provides the `symfind` utility to assist you in locating a library or object file where a symbol is defined. When linking a program, undefined symbols are a common error that produce an error message similar to the following:

```
$ make -f demo_proc.mk sample1
Undefined                       first referenced
 symbol                             in file
sqlcex                              sample1.o
sqlglm                              sample1.o
ld: fatal: Symbol referencing errors. No output written to sample1
```

The error occurs when the linker cannot find a definition for a referenced symbol. If this error message occurs, ensure that the library or object file containing the definition exists on the link line and that the linker is searching the correct directories for the file.

The following example shows the output from the `symfind` utility, used to locate the `sqlcex` symbol:

```
$ symfind sqlcex

SymFind - Find Symbol <sqlcex> in <**>.a, .o, .so
-------------------------------------------------
Command:         /u01/app/oracle/product/10.1.0/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/10.1.0
Output File:     (none)
Note:            I do not traverse symbolic links
                 Use '-v' option to show any symbolic links

Locating Archive and Object files ...
[11645] |    467572|      44|FUNC |GLOB |0    |8      |sqlcex
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ./lib/libclntsh.sl
[35]    |         0|      44|FUNC |GLOB |0    |5      |sqlcex
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ./lib/libsql.a
```

Intimate Shared Memory

On Solaris systems, Oracle Database uses Intimate Shared Memory (ISM) for shared memory segments because it shares virtual memory resources between Oracle processes. ISM causes the physical memory for the entire shared memory segment to be locked automatically.

On Solaris 8 and Solaris 9 systems, dynamic/pageable ISM (DISM) is available. This enables Oracle Database to share virtual memory resources between processes sharing the segment, and at the same time, enables memory paging. The operating system does not have to lock down physical memory for the entire shared memory segment.

Oracle Database automatically decides at startup whether to use ISM or DISM, based on the following criteria:

- Oracle Database uses DISM if it is available on the system, and if the value of the SGA_MAX_SIZE initialization parameter is larger than the size required for all SGA components combined. This allows Oracle Database to lock only the amount of physical memory that is used.
- Oracle Database uses ISM if the entire shared memory segment is in use at startup or if the value of the SGA_MAX_SIZE parameter is equal to or smaller than the size required for all SGA components combined.

Regardless of whether Oracle Database uses ISM or DISM, it can always exchange the memory between dynamically sizable components such as the buffer cache, the shared pool, and the large pool after it starts an instance. Oracle Database can relinquish memory from one dynamic SGA component and allocate it to another component.

Because shared memory segments are not implicitly locked in memory, when using DISM, Oracle Database explicitly locks shared memory that is currently in use at startup. When a dynamic SGA operation uses more shared memory, Oracle Database explicitly performs a lock operation on the memory that comes in use. When a dynamic SGA operation releases shared memory, Oracle Database explicitly performs an unlock operation on the memory that is freed, so that it becomes available to other applications.

Oracle Database uses the `oradism` utility to lock and unlock shared memory. The `oradism` utility is automatically set up during installation. You do not need to perform any configuration tasks to use dynamic SGA.

Note: Oracle recommends that you do not set the LOCK_SGA parameter to TRUE in the server parameter file on Solaris systems. If you do, Oracle Database 10*g* does not start up.

Note: The process name for the `oradism` utility is `ora_dism_`*`sid`*, where *`sid`* is the system identifier. When using DISM, this process is started during instance startup, and automatically quits when the instance is shut down.

If a message appears in the alert log saying that the `oradism` utility is not set up correctly, verify that the `oradism` utility is located in the `$ORACLE_HOME/bin` directory and that it has superuser privileges.

F

Database Limits

This appendix describes database limits specific to UNIX systems.

Database Limits

Table F–1 lists the default and maximum values for parameters in a CREATE DATABASE or CREATE CONTROLFILE statement.

Note: Interdependencies between these parameters might affect allowable values.

Table F–1 CREATE CONTROLFILE and CREATE DATABASE Parameters

Parameter	Default	Maximum Value
MAXLOGFILES	16	255
MAXLOGMEMBERS	2	5
MAXLOGHISTORY	100	65534
MAXDATAFILES	30	65534
MAXINSTANCES	1	63

Table F–2 lists the Oracle Database file size limits in bytes specific to UNIX.

Table F–2 File Size Limits

File Type	Operating System	Maximum SIze
Data files	Any	4,194,303 multiplied by the value of the DB_BLOCK_SIZE parameter
Import/Export files and SQL*Loader files	Tru64 UNIX	16 TB
	AIX, HP-UX, Linux, Solaris: 32-bit with 32-bit files	2,147,483,647 bytes
	AIX, HP-UX, Linux, Solaris: 32-bit with 64-bit files	Unlimited
	AIX, HP-UX, Linux, Solaris: 64-bit	Unlimited
Control files	Solaris, HP-UX, Linux	20000 database blocks
	AIX	10000 database blocks
	Tru64 UNIX	19200 database blocks

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

X