Oracle® Database

Administrator's Reference

10*g* Release 2 (10.2) for UNIX-Based Operating Systems

B15658-03

September 2005

ORACLE®

Oracle Database Administrator's Reference, 10*g* Release 2 (10.2) for UNIX-Based Operating Systems

B15658-03

Primary Authors: Apolina Das, Sanjay Sharma, Lyju Vadassery

Contributing Authors: Kevin Flood, Pat Huey, Clara Jaeckel, Emily Murphy, Terri Winters

Contributors: David Austin, Subhranshu Banerjee, Mark Bauer, Robert Chang, Jonathan Creighton, Sudip Datta, Padmanabhan Ganapathy, Thirumaleshwara Hasandka, Joel Kallman, George Kotsovolos, Richard Long, Rolly Lv, Padmanabhan Manavazhi, Matthew Mckerley, Sreejith Minnanghat, Krishna Mohan, Douglas Williams, Rajendra Pingte, Hanlin Qian, Janelle Simmons, Preeti Shukla, Roy Swonger

Contents

2 Stopping and Starting Oracle Software

3 Configuring Oracle Database

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 Oracle Database

A Administering Oracle Database on AIX

E Administering Oracle Database on Solaris

F Administering Oracle Database on Tru64 UNIX

G Using Oracle ODBC Driver

Preface

This guide provides information about administering and configuring Oracle Database 10*g* release 2 (10.2) on the following platforms:

- AIX 5L Based Systems (64-Bit)
- hp-ux PA-RISC (64-Bit)
- Linux x86
- Linux x86-64
- Solaris Operating System (SPARC 64-Bit)

Audience

This guide is intended for anyone responsible for administering and configuring Oracle Database 10*g* release 2 (10.2). If you are configuring Oracle Real Application Clusters (RAC), then refer to *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide*.

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

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

Related Documentation

Refer to the appropriate section for a listing of Oracle Database 10*g* documentation specific to your platform:

- AIX 5L Based Systems (64-Bit) Documentation
- hp-ux PA-RISC (64-Bit) Documentation
- Linux x86 Documentation
- Linux x86-64 Documentation
- Solaris Operating System (SPARC 64-Bit) Documentation

AIX 5L Based Systems (64-Bit) Documentation

- Oracle Database
  - *Oracle Database Release Notes for AIX 5L Based Systems (64-Bit)*
  - *Oracle Database Installation Guide for AIX 5L Based Systems (64-Bit)*
  - *Oracle Database Quick Installation Guide for AIX 5L Based Systems (64-Bit)*
  - *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Installation Guide for AIX Based Systems*
  - *Oracle Database Administrator's Reference for UNIX-Based Operating Systems*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for AIX 5L Based Systems (64-Bit)*
  - *Oracle Database Client Quick Installation Guide for AIX 5L Based Systems (64-Bit)*
- Oracle Database 10*g* Companion CD
  - *Oracle Database Companion CD Installation Guide for AIX 5L Based Systems (64-Bit)*
  - *Oracle Database Companion CD Quick Installation Guide for AIX 5L Based Systems (64-Bit)*

hp-ux PA-RISC (64-Bit) Documentation

- Oracle Database
  - *Oracle Database Release Notes for hp-ux PA-RISC (64-Bit)*
  - *Oracle Database Installation Guide for hp-ux PA-RISC (64-Bit)*
  - *Oracle Database Quick Installation Guide for hp-ux PA-RISC (64-Bit)*
  - *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Installation Guide for hp-ux*

  - *Oracle Database Administrator's Reference for UNIX-Based Operating Systems*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for hp-ux PA-RISC (64-Bit)*
  - *Oracle Database Client Quick Installation Guide for hp-ux PA-RISC (64-Bit)*
- Oracle Database 10*g* Companion CD
  - *Oracle Database Companion CD Installation Guide for hp-ux PA-RISC (64-Bit)*
  - *Oracle Database Companion CD Quick Installation Guide for hp-ux PA-RISC (64-Bit)*

Linux x86 Documentation

- Oracle Database
  - *Oracle Database Release Notes for Linux x86*
  - *Oracle Database Installation Guide for Linux x86*
  - *Oracle Database Quick Installation Guide for Linux x86*
  - *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Installation Guide for Linux*
  - *Oracle Database Administrator's Reference for UNIX-Based Operating Systems*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for Linux x86*
  - *Oracle Database Client Quick Installation Guide for Linux x86*
- Oracle Database 10*g* Companion CD
  - *Oracle Database Companion CD Installation Guide for Linux x86*
  - *Oracle Database Companion CD Quick Installation Guide for Linux x86*

Linux x86-64 Documentation

- Oracle Database
  - *Oracle Database Release Notes for Linux x86-64*
  - *Oracle Database Installation Guide for Linux x86-64*
  - *Oracle Database Quick Installation Guide for Linux x86-64*
  - *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Installation Guide for Linux*
  - *Oracle Database Administrator's Reference for UNIX-Based Operating Systems*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for Linux x86-64*
  - *Oracle Database Client Quick Installation Guide for Linux x86-64*
- Oracle Database 10*g* Companion CD
  - *Oracle Database Companion CD Installation Guide for Linux x86-64*
  - *Oracle Database Companion CD Quick Installation Guide for Linux x86-64*

Solaris Operating System (SPARC 64-Bit) Documentation

- Oracle Database
  - *Oracle Database Release Notes for Solaris Operating System (SPARC 64-Bit)*
  - *Oracle Database Installation Guide for Solaris Operating System (SPARC 64-Bit)*
  - *Oracle Database Quick Installation Guide for Solaris Operating System (SPARC 64-Bit)*
  - *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Installation Guide for Solaris Operating System*
  - *Oracle Database Administrator's Reference for UNIX-Based Operating Systems*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for Solaris Operating System (SPARC 64-Bit)*
  - *Oracle Database Client Quick Installation Guide for Solaris Operating System (SPARC 64-Bit)*
- Oracle Database 10*g* Companion CD
  - *Oracle Database Companion CD Installation Guide for Solaris Operating System (SPARC 64-Bit)*
  - *Oracle Database Companion CD Quick Installation Guide for Solaris Operating System (SPARC 64-Bit)*

For important information that was not available when this book was released, refer to the release notes for your platform. The release notes for Oracle Database are updated regularly. You can get the most recent version from Oracle Technology Network at

`http://www.oracle.com/technology/documentation/index.html`

Conventions

The following text conventions are used in this document:

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

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]`

Terminology

The names of some UNIX operating systems have been shortened in this guide. These are:

Operating System	Abbreviated Name
AIX 5L Based Systems (64-Bit)	AIX
Apple Mac OS X	Mac OS X
hp-ux PA-RISC (64-bit) hp-ux Itanium	HP-UX **Note:** Where the information for HP-UX is different on a particular architecture, this is noted in the text.
hp Tru64 UNIX	Tru64 UNIX
Linux Itanium Linux on POWER IBM zSeries Based Linux Linux x86 Linux x86-64	Linux **Note:** Where the information for Linux is different on a particular architecture, this is noted in the text.
Solaris Operating System (SPARC 64-Bit) Solaris Operating System (x86) Solaris Operating System (x86-64)	Solaris **Note:** Where the information for Solaris is different on a particular architecture, this is noted in the text.

Accessing Documentation

The documentation for Oracle Database 10*g* release 2 (10.2) 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 DVDs 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 Oracle Database 10*g* Release 2 (10.2) Online Documentation Library. This library is on the Oracle Technology Network Web site at

`http://www.oracle.com/technology/documentation/index.html`

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 also 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 Database

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

> **See Also:** The appropriate appendix in this guide for platform-specific information about administering Oracle Database

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.

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 at sign (@) represents the `ORACLE_SID` environment variable. For example, to indicate a file belonging to the current instance, run the following command:

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

Environment Variables

This section describes the most commonly used Oracle Database and operating system environment variables. You must define some of these environment variables before installing Oracle Database.

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, run the following command:

```
$ env | grep ORACLE_SID
```

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

```
$ env | more
```

Oracle Database Environment Variables

Table 1–1 describes the environment variables used with Oracle Database.

Table 1–1 Oracle Database Environment Variables

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 the character set is automatically converted. Refer to *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 about 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/10.2.0/db_1
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, then the Oracle application uses ORACLE_PATH to locate the file.
	Syntax	Colon-separated list of directories: *directory1*:*directory2*:*directory3*
	Example	/u01/app/oracle/product/10.2.0/db_1/bin:. **Note:** The period adds the current working directory to the search path.
ORACLE_SID	Function	Specifies the Oracle system identifier.

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

Variable	Detail	Definition
	Syntax	A string of numbers and letters that must begin with a letter. Oracle recommends a maximum of 8 characters for system identifiers. For more information about this environment variable, refer to *Oracle Database Installation Guide for Linux x86*.
	Example	SAL1
ORACLE_TRACE	Function	Enables the tracing of shell scripts during an installation. If it is set to T, then 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, then the scripts do not use the set -x command.
	Syntax	T or not T
	Example	T
ORAENV_ASK	Function	Controls whether the oraenv or coraenv script prompts or does not prompt for the value of the ORACLE_SID environment variable. If it is set to NO, then 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, then 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
TWO_TASK	Function	Specifies the default connect identifier to use in the connect string. If this environment variable is set, then do not specify the connect identifier in the connect string. For example, if the TWO_TASK environment variable is set to sales, then you can connect to a database by 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 by 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 Database server processes, for example ARCH, PMON, and DBWR.

UNIX Environment Variables

Table 1–2 describes UNIX environment variables used with Oracle Database.

Table 1–2 Environment Variables Used with Oracle Database

Variable	Detail	Definition
ADA_PATH (AIX only)	Function	Specifies the directory containing the Ada compiler.sm
	Syntax	*directory_path*
	Example	/usr/lpp/powerada
CLASSPATH	Function	Used with Java applications. The required setting for this variable depends on the Java application. Refer to 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
DISPLAY	Function	Used by X-based tools. Specifies the display device used for input and output. Refer to the 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 use a single monitor, then use the value 0 for both server and screen (0.0). **Note:** If you use a single monitor, then *screen* is optional.
	Example	135.287.222.12:0.0 bambi:0
DYLD_LIBRARY_PATH (Mac OS X only)	Function	Specifies the list of directories that the shared library loader searches to locate shared object libraries at run time. See the dyld man page for information about this environment variable.
	Syntax	Colon-separated list of directories: *directory1:directory2:directory3*
	Example	/usr/lib:$ORACLE_HOME/lib
HOME	Function	The home directory of the user.
	Syntax	*directory_path*
	Example	/home/oracle
LANG or LANGUAGE	Function	Specifies the language and character set used by the operating system for messages and other output. Refer to the operating system documentation for more information. **Note:** This environment variable is not used on Apple Mac OS X.
LD_OPTIONS	Function	Specifies the default linker options. Refer to the ld man page for more information about this environment variable.
LPDEST (Solaris only)	Function	Specifies the name of the default printer.
	Syntax	*string*
	Example	docprinter
LD_LIBRARY_PATH (All platforms except AIX and Mac OS X.)	Function	Specifies the list of directories that the shared library loader searches to locate shared object libraries at run time. Refer to the ld man page for information about this environment variable. On HP-UX, specifies the path for 64-bit shared libraries.
	Syntax	Colon-separated list of directories: *directory1:directory2:directory3*
	Example	/usr/dt/lib:$ORACLE_HOME/lib
LD_LIBRARY_PATH_64 (SPARC systems only)	Function	Specifies the list of directories that the shared library loader searches to locate specific 64-bit shared object libraries at run time. Refer to 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 (Cont.) Environment Variables Used with Oracle Database

Variable	Detail	Definition
LIBPATH (AIX only)	Function	Specifies the list of directories that the shared library loader searches to locate shared object libraries at run time. Refer to 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:$HOME/bin:. **Note**: The period adds the current working directory to the search path.
PRINTER	Function	Specifies the name of the default printer.
	Syntax	*string*
	Example	docprinter
SHLIB_PATH (HP-UX 32-bit libraries only)	Function	Specifies the list of directories that the shared library loader searches to locate shared object libraries at run time. Refer to the ld man page for information about this environment variable.
	Syntax	Colon-separated list of directories: *directory1*: *directory2*: *directory3*
	Example	/usr/dt/lib:$ORACLE_HOME/lib32
TEMP, TMP, and TMPDIR	Function	Specifies 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. Refer to the X Window System documentation for more information.

Setting a Common Environment

This section describes how to set a common operating system environment by 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 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 may find yourself frequently adding and removing databases from your development system or your users may 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, use the appropriate command:

- `coraenv` script:

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

- `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 must have read access to this directory. Include the path of the local bin directory `PATH` environment variable setting for the users. 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, then 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 value of `SYSDATE`.

Oracle recommends that you do not change your personal TZ value. Using different values of TZ, such as GMT+24, may 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 information about Oracle Database initialization parameters:

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 is 16 KB on Linux, Mac, and Solaris. It is 32 KB on AIX, HP-UX, and Tru64 UNIX.

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

ASM_DISKSTRING Initialization Parameter

Note: Only Automatic Storage Management instances support the ASM_DISKSTRING initialization parameter.

The syntax for assigning a value to the ASM_DISKSTRING initialization parameter is as follows:

```
ASM_DISKSTRING = 'path1'[,'path2', . . .]
```

In this syntax, *pathn* is the path to a raw device. You can use wildcard characters when specifying the path.

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

Table 1–3 Default Values of the ASM_DISKSTRING Initialization Parameter

Platform	Default Search String
AIX	/dev/rhdisk*
HP-UX	/dev/rdsk/*
Linux	/dev/raw/*
Mac OS X	/dev/rdisk*s*s1
Solaris	/dev/rdsk/*
Tru64 UNIX	/dev/rdisk/*

See Also: The glob(7) man page for platform-specific information about the wildcard character patterns that you can use when specifying the path of a raw device

LOG_ARCHIVE_DEST_n Initialization Parameter

The maximum value that you can set for ASYNC in the LOG_ARCHIVE_DEST_*n* initialization parameter differs on UNIX platforms as listed in the following table.

Platform	Maximum Value
zSeries Linux	12800
HP-UX and Tru64 UNIX	51200
Other operating systems	102400

Operating System Accounts and Groups

This section describes the following special operating system accounts and groups that are required by Oracle Database:

- Oracle Software Owner Account
- OSDBA, OSOPER, and Oracle Inventory Groups
- Groups and Security
- External Authentication
- Running the orapwd Utility
- Password Management
- Creating Additional Operating System Accounts
- Configuring the Accounts of Oracle Users

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 have 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 operating system groups required by Oracle Database.

Table 1–4 Operating System 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 must belong to the same operating system group. This group is called the Oracle Inventory group. It must be the primary group of the Oracle software owner during installations. After the installation, this group owns all the Oracle files installed on the system.

See Also: *Oracle Database Administrator's Guide* and *Oracle Database Installation Guide for Linux x86* for more information about the OSDBA group and SYSDBA privileges, and the OSOPER group and SYSOPER privileges

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 during processing.

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:** *Oracle Database Administrator's Guide* for more information about security issues

Groups and Security

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

- The primary group for the `oracle` account must 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, the `oracle` user.

External Authentication

If you choose to use external authentication, then 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, then 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:

```
OS_AUTHENT_PREFIX=""
```

> **See Also:** *Oracle Database Administrator's Guide* for more information about external authentication

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 Oracle Database Configuration Assistant to create a database, then the assistant creates a password file for the new database. If you create the database manually, then 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 in which password information is written The name of the file must be `orapwsid`, and you must supply the full path name. Its contents are encrypted. Typically, the password file is 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, then 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 permitted in the password file. This is the maximum number of distinct users permitted to connect to the database simultaneously with either the SYSDBA or the SYSOPER privilege.

See Also: *Oracle Database Administrator's Guide* for more information about using the `orapwd` utility

Password Management

When using Oracle 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, Oracle 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 using them. To do this, you can use one of the following methods:

- To change the passwords by using Oracle Database Configuration Assistant, click **Password Management** in the Database Configuration Assistant Summary window.
- Alternatively, use SQL*Plus to connect to the database as SYS and run the following command to unlock an account and reset its password:

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

Creating Additional Operating System Accounts

If required, create additional operating system 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 operating system 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 the `.bash_profile` file for the Bash shell on Red Hat Enterprise Linux and Mac.

For the C 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):

- Guidelines for Using Raw Devices
- Raw Device Setup
- Raw Device Data Files on AIX and Tru64 UNIX Systems
- Raw Device Support on Linux Systems

See Also: The appendix corresponding to your platform in this guide for additional raw device tuning information

Guidelines for Using Raw Devices

Raw devices (raw partitions or raw volumes) have the following potential disadvantages:

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

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

  – Bourne, Bash, or Korn shell:

  ```
  $ ulimit -a
  ```

  – C shell:

  ```
  % limit
  ```

- If a particular disk drive has intense I/O activity and performance would benefit from movement of an Oracle data file to another drive, then it is likely that no acceptably sized partition or volume exists on a drive with less I/O activity. It may 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 disk group.

Consider the following issues when deciding to use raw devices:

- 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, then 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 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
  - Enabling 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 system, as well as those that can be shared with more than one system of a cluster. Shared drives enables 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 low activity. Most hardware vendors who provide the logical disk facility also provide a graphical user interface (GUI) 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, then dynamic resynchronization is possible. Some vendors also provide the ability to replace drives online in conjunction with the mirroring facility.

Raw Device Setup

When creating raw 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 is at least two Oracle block sizes smaller than the size of the raw partition.

> **See Also:** The operating system documentation for more information about creating raw devices

Raw Device Data Files on AIX and Tru64 UNIX Systems

On AIX and Tru64 UNIX systems, data files 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. You may need to do this, for example, if you want to transfer the data file to a different device.

> **See Also:** Appendix A for more information about creating a raw logical volume on AIX, which enables you to use a zero offset. The zero offset is recommended for raw logical volumes on AIX.

Raw Device Support on Linux Systems

You can use both raw character devices and block devices as raw volumes in creating a database. Because block devices are supported, the kernel level limitation on the maximum number of raw devices is removed and you can configure additional raw volumes to meet your requirements.

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. It includes the following sections:

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 operating system process ID number

The following is a sample trace file name:

```
$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, then 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 this initialization parameter, then 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, then the default directory is $ORACLE_HOME/rdbms/log.

2

Stopping and Starting Oracle Software

This chapter describes how to identify Oracle Database processes, and provides basic information about how to stop and restart them. It also describes how to set up automatic startup and shutdown of the Oracle Database. It contains the following sections:

- Stopping and Starting Oracle Processes
- Automating Shutdown and Startup

Stopping and Starting Oracle Processes

This section describes how to stop and start Oracle processes. It contains the following topics:

- Starting Oracle Processes on Mac OS X
- Stopping and Starting Oracle Database and Automatic Storage Management Instances
- Stopping and Starting Oracle CSS Daemon
- Stopping and Starting an Oracle Net Listener
- Stopping and Starting iSQL*Plus
- Stopping and Starting Oracle Ultra Search
- Stopping and Starting Oracle Enterprise Manager Database Control
- Stopping and Starting Oracle Management Agent

Starting Oracle Processes on Mac OS X

Note: Ensure that you follow the instructions in this section every time you start an Oracle Database or Automatic Storage Management instance or an Oracle Net listener process.

To ensure that certain shell limits are set to the values required to run Oracle processes, you must use the `ssh`, `rlogin`, or `telnet` command to connect to the system where you want to start the process, even if that system is the local system. The syntax of this command is as follows:

```
$ ssh localhost
```

Stopping and Starting Oracle Database and Automatic Storage Management Instances

This section describes how to stop and start Oracle Database and Automatic Storage Management instances.

Stopping an Oracle Database or Automatic Storage Management Instance

> **Caution:** Do not stop an Automatic Storage Management instance until you have stopped all Oracle Database instances that use that Automatic Storage Management instance to manage their storage.

To stop an Oracle Database or Automatic Storage Management instance:

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

   On Solaris:

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

   On 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 Automatic Storage Management 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 Automatic Storage Management instances.

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:

   - Bourne, Bash, or Korn shell:

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

   - C shell:

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

   When prompted, specify the SID for the instance.

3. Run 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 quit SQL*Plus.

Restarting an Oracle Database or Automatic Storage Management Instance

Caution: If the database instance uses Automatic Storage Management for storage management, then you must start the Automatic Storage Management instance before you start the database instance.

To restart an Oracle Database or Automatic Storage Management instance:

1. If required, repeat Steps 1 and 2 of the preceding procedure 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.
2. Run 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.

Stopping and Starting Oracle CSS Daemon

To stop the Oracle Cluster Services Synchronization (CSS) daemon, run the following command:

On AIX and Mac OS X:

```
/etc/init.cssd stop
```

On other platforms:

```
/etc/init.d/init.cssd stop
```

To start the CSS daemon, run the following command:

```
$ORACLE_HOME/bin/localconfig reset
```

This command stops the Oracle CSS daemon and then restarts it.

Stopping and Starting an Oracle Net Listener

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

Stopping Oracle Net Listener

To stop an Oracle Net listener:

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

   On Mac OS X:

   ```
   $ ps -auxwww | grep tnslsnr
   ```

   On other platforms:

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

   This command displays a list of the Oracle Net listeners running on the system. The output of this command is similar to the following:

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

In this sample output, *listenername1* and *listenername2* are the names of the listeners.

2. If required, set the ORACLE_HOME environment variable to specify the appropriate Oracle home directory for the listener that you want to stop:
   - Bourne, Bash, or Korn shell:

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

   - C shell:

     ```
     % setenv ORACLE_HOME oracle_home1
     ```

3. Run 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, then you do not have to specify the name in this command.

Restarting Oracle Net Listener

To start an Oracle Net listener:

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

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

   - C shell:

     ```
     % setenv ORACLE_HOME oracle_home1
     ```

2. Run 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. The listener name is mentioned in the listener.ora file. To display the contents of this file, run the following command:

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

Stopping and Starting *i*SQL*Plus

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

Stopping *i*SQL*Plus

To stop *i*SQL*Plus:

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

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

- C shell:

```
% setenv ORACLE_HOME oracle_home
```

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

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

Starting *i*SQL*Plus

To start *i*SQL*Plus:

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

   - Bourne, Bash, or Korn shell:

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

   - C shell:

```
% setenv ORACLE_HOME oracle_home
```

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

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

Stopping and Starting Oracle Ultra Search

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

Stopping Oracle Ultra Search

To stop Oracle Ultra Search:

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

   - Bourne, Bash, or Korn shell:

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

   - C shell:

```
% setenv ORACLE_HOME oracle_home
```

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

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

Starting Oracle Ultra Search

To start Oracle Ultra Search:

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

   - Bourne, Bash, or Korn shell:

```
$ ORACLE_HOME=oracle_home
```

```
$ export ORACLE_HOME
```

- C shell:

```
% setenv ORACLE_HOME oracle_home
```

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

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

Stopping and Starting Oracle Enterprise Manager Database Control

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

Stopping Oracle Enterprise Manager Database Control

To stop Oracle Enterprise Manager Database Control:

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:

   - `coraenv` script:

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

   - `oraenv` script:

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

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

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

Starting Oracle Enterprise Manager Database Control

To start Database Control:

1. 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:

   - Bourne, Bash, or Korn shell:

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

   - C shell:

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

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

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

Stopping and Starting Oracle Management Agent

If you are using Oracle Enterprise Manager Grid Control to manage multiple Oracle products from a central location, then 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:

1. Run the following command to determine the Oracle home directory for Oracle Management Agent:

   On Mac OS X:

   ```
   $ ps -auxwww | grep emagent
   ```

   On other platforms:

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

   This command displays information about the Oracle Management Agent processes. The output of this command is similar to the following:

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

2. If required, set the `ORACLE_HOME` environment variable to specify the appropriate Oracle home directory for the Oracle Management Agent:

   - Bourne, Bash, or Korn shell:

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

   - C shell:

     ```
     % setenv ORACLE_HOME oracle_home
     ```

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

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

Starting Oracle Management Agent

To start Oracle Management Agent:

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

   - Bourne, Bash, or Korn shell:

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

   - C shell:

     ```
     % setenv ORACLE_HOME oracle_home
     ```

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

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

Automating Shutdown and Startup

Oracle recommends that you configure your system to automatically start Oracle Database when the system starts up, and to automatically shut it 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, which are located in the `$ORACLE_HOME/bin` directory. The scripts refer to the same

entries in the oratab file, which are applied on 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 do not include the dbstart entry from the system startup files.

> **See Also:** The init command in your operating system documentation for more information about system startup and shutdown procedures

- Automating Database Shutdown and Startup on Mac OS X
- Automating Database Startup and Shutdown on Other Operating Systems

Automating Database Shutdown and Startup on Mac OS X

To automate database startup and shutdown by using the dbstart and dbshut scripts:

1. Log in as the root user.
2. Open the oratab file in any text editor:

   ```
   # vi /etc/oratab
   ```

   Database entries in the oratab file are displayed 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 for which you want to automate shutdown and startup, first find the instance identifier (SID) for that database, which is identified by the *SID* in the first field. Then, change the last field for each to Y.

   > **Note:** If you add new database instances to the system, then remember to edit the entries for those instances in the oratab file if you want them to start automatically.

3. Run the following commands to create the /Library/StartupItems/Oracle directory and to change directory to it:

   ```
   # mkdir /Library/StartupItems/Oracle
   # cd /Library/StartupItems/Oracle
   ```

4. Using any text editor, create a startup script called Oracle in this directory, with contents similar to the following:

   ```
   #!/bin/bash

   # source the common startup script

   . /etc/rc.common

   # Change the value of ORACLE_HOME to specify the correct Oracle home
   # directory for the installation

   ORACLE_HOME=/Volumes/u01/app/oracle/product/10.2.0/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
export ORACLE_HOME PATH

# Set shell limits for the Oracle Database

ulimit -c unlimited
ulimit -d unlimited
ulimit -s 65536

StartService()
{
        if [ -f $ORACLE_HOME/bin/tnslsnr ] ; then
             ConsoleMessage "Starting Oracle Net"
             su $ORACLE -c "$ORACLE_HOME/bin/lsnrctl start"
        fi
        ConsoleMessage "Starting Oracle Databases"
        su $ORACLE -c "$ORACLE_HOME/bin/dbstart $ORACLE_HOME"
}

StopService()
{
        ConsoleMessage "Stopping Oracle Databases"
        su $ORACLE -c "$ORACLE_HOME/bin/dbshut $ORACLE_HOME"
        if [ -f $ORACLE_HOME/bin/tnslsnr ] ; then
             ConsoleMessage "Stopping Oracle Net"
             su $ORACLE -c "$ORACLE_HOME/bin/lsnrctl stop"
        fi
}

RestartService()
{
       StopService
       StartService
}

RunService "$1"
```

Note: The script can only stop Oracle Net listener for which a password has not been set. In addition, if the listener name is not the default name `LISTENER`, then you must specify the listener name in the `stop` and `start` commands:

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

---

5. Using any text editor, create a startup item parameter list file called `StartupParameters.plist` in this directory, with contents similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
```

```
    <key>Description</key>
    <string>Oracle Database Startup</string>
    <key>Provides</key>
    <array>
       <string>Oracle Database</string>
    </array>
    <key>Requires</key>
    <array>
       <string>Disks</string>
    </array>
    <key>Uses</key>
    <array>
       <string>Disks</string>
       <string>Network</string>
       <string>NFS</string>
    </array>
    <key>OrderPreference</key>
    <string>Late</string>
</dict>
</plist>
```

Caution: AL32UTF8 is the Oracle Database character set that is appropriate for XMLType data. It is equivalent to the IANA registered standard UTF-8 encoding, which supports all valid XML characters.

Do not confuse the Oracle Database database character set UTF8 (no hyphen) with the database character set AL32UTF8 or with character encoding UTF-8. Database character set UTF8 has been superseded by AL32UTF8. Do not use UTF8 for XML data. UTF8 supports only Unicode version 3.1 and earlier; it does not support all valid XML characters. AL32UTF8 has no such limitation.

Using database character set UTF8 for XML data could potentially cause a fatal error or affect security negatively. If a character that is not supported by the database character set appears in an input-document element name, a replacement character (usually "?") is substituted for it. This will terminate parsing and raise an exception.

---

6. Change the owner, group, and permissions of the files that you created as follows:

```
# chown root:wheel *
# chmod 700 *
```

Automating Database Startup and Shutdown on Other Operating Systems

To automate database startup and shutdown by using the `dbstart` and `dbshut` scripts:

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

   To open the file, use one of the following commands:

   - On Solaris:

     ```
     # vi /var/opt/oracle/oratab
     ```

   - On AIX, HP-UX, Linux, and Tru64 UNIX:

     ```
     # vi /etc/oratab
     ```

Database entries in the `oratab` file are displayed in the following format:

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

In this example, the values `Y` and `N` specify whether you want the scripts to start up or shut down the database, respectively. For each database for which you want to automate shutdown and startup, first determine the instance identifier (SID) for that database, which is identified by the *SID* in the first field. Then, change the last field for each to `Y`.

You can set `dbstart` to auto-start a single-instance database that uses an Automatic Storage Management installation that is auto-started by Oracle Clusterware. This is the default behavior for an Automatic Storage Management cluster. If you want to do this, then you must change the `oratab` entry of the database and the Automatic Storage Management installation to use a third field with the value `W` and `N`, respectively. These values specify that `dbstart` auto-starts the database only after the Automatic Storage Management instance is started.

> **Note:** If you add new database instances to the system and if you want to automate startup for them, then you must edit the entries for those instances in the `oratab` file.

3. Change directory to one of the following depending on your operating system.

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`, and copy the following lines into this file:

> **Note:** Change the value of the `ORACLE_HOME` environment variable to an Oracle home directory for the installation. Change the value of the `ORACLE` environment variable to the user name 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 your installation.

ORACLE_HOME=/u01/app/oracle/product/10.2.0/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
#
case $1 in
'start')
        $ORACLE_HOME/bin/dbstart $ORACLE_HOME &
        ;;
'stop')
        $ORACLE_HOME/bin/dbshut $ORACLE_HOME &
        ;;
*)
        echo "usage: $0 {start|stop}"
        exit
        ;;
esac
#
exit
```

> **Note:** This script can only stop Oracle Net listener for which a password has not been set. In addition, if the listener name is not the default name, `LISTENER`, then 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`

Platform	Symbolic Links Commands
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 Database

This chapter describes how to configure Oracle Database for Oracle products. It contains the following sections:

Configuring Oracle Database for Additional Oracle Products

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

1. If required, start the database.
2. Run the following command to start Oracle 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. Select the products that you want to enable, and 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 start each of the assistants noninteractively by using a response file.

This section contains the following topics:

Using Oracle Net Configuration Assistant

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

If you choose to perform a separate Oracle Database Client installation, then Oracle Net Configuration Assistant automatically creates a configuration that is consistent with the selections made during the installation. Oracle Universal Installer automatically runs 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 Oracle Net Configuration Assistant to create a more detailed configuration by entering the following command:

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

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

Using Oracle Database Upgrade Assistant

During an Oracle Database installation, you can choose 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, then you can run Oracle Database Upgrade Assistant after the installation.

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

To start Oracle Database Upgrade Assistant, run the following command:

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

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

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

See Also: *Oracle Database Installation Guide for Linux x86* and *Oracle Database Upgrade Guide* for more information about upgrades

Using Oracle Database Configuration Assistant

You can use Oracle 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 Oracle Database Configuration Assistant, run the following command:

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

For information about the command-line options available with Oracle 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 may 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:

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

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

   - C shell:

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

   When prompted, specify the SID for the database.
3. Run the following command to start SQL*Plus:

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

4. If required, run the following command to 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 by 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.

> **Note:** Before relinking executables, you must shut down all executables that are running in the Oracle home directory that you are relinking. In addition, shut down applications linked with Oracle shared libraries.

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

To relink product executables, run the following command:

```
$ relink argument
```

In this command, *argument* is one of the values listed in Table 3–1.

Table 3–1 Relink Script Arguments

Argument	Description
`all`	Every product executable that has been installed
`oracle`	Oracle Database executable only

Table 3–1 (Cont.) Relink Script Arguments

Argument	Description
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
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 administer SQL*Plus. It contains the following sections:

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

> **See Also:** *SQL*Plus User's Guide and Reference* for more information about SQL*Plus

Administering Command-Line SQL*Plus

This section describes how to administer command-line SQL*Plus. In the examples, SQL*Plus replaces the question mark (?) with the value of the `ORACLE_HOME` environment variable.

- Using Setup Files
- Using the PRODUCT_USER_PROFILE Table
- Using Oracle Database Sample Schemas
- Installing and Removing SQL*Plus Command-Line Help

Using Setup Files

When you start SQL*Plus, it runs the `glogin.sql` site profile setup file and then runs the `login.sql` user profile setup 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, then it is overwritten when you install SQL*Plus. If SQL*Plus is removed, then 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. The value of this environment variable is a colon-separated list of directories. SQL*Plus searches these directories for the `login.sql` file in the order that they are listed in the `SQLPATH` environment variable.

The options set in the `login.sql` file override those set in the `glogin.sql` file.

> **See Also:** *SQL*Plus User's Guide and Reference* for more information about profile files

Using the PRODUCT_USER_PROFILE Table

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

> **See Also:** *Oracle Database Installation Guide for Linux x86* for more information about installation options

To re-create the PRODUCT_USER_PROFILE table, run the `$ORACLE_HOME/sqlplus/admin/pupbld.sql` script in the SYSTEM schema. For example, run 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 re-create 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 want to run the `pupbld` script without interaction, then set the `SYSTEM_PASS` environment variable to the SYSTEM user name and password.

Using Oracle Database Sample Schemas

When you install Oracle Database or use Oracle Database Configuration Assistant to create a database, you can choose to install Oracle Database Sample Schemas.

> **See Also:** *Oracle Database Sample Schemas* for information about installing and using Oracle Database Sample Schemas

Installing and Removing SQL*Plus Command-Line Help

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

> **See Also:** *SQL*Plus User's Guide and Reference* for more information about the SQL*Plus command-line Help

Installing 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 by using the `$ORACLE_HOME/bin/helpins` shell script.

  The `helpins` script prompts for the SYSTEM password. If you want to run this script without interaction, then 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 shell:

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

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

  For example, run 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 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. To do this, run 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. It contains the following topics:

- Using a System Editor from SQL*Plus
- Running Operating System Commands from SQL*Plus
- Interrupting SQL*Plus
- Using the SPOOL Command

Using a System Editor from SQL*Plus

If you run an `ED` or `EDIT` command at the SQL*Plus prompt, then 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 should start by defining the SQL*Plus `_EDITOR` variable. You can define this variable in the `glogin.sql` site profile or the `login.sql` user profile. Alternatively, you can define it during the SQL*Plus session. For example, to set the default editor to `vi`, run the following command:

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

If you do not set the `_EDITOR` variable, then the value of either the `EDITOR` or the `VISUAL` environment variable is used. If both environment variables are set, then the

value of the EDITOR variable is used. If _EDITOR, EDITOR, and VISUAL are not specified, then the default editor is ed.

When 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:

```
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 point (!) as the first character after the SQL*Plus prompt causes subsequent characters to be passed to a subshell. The SHELL environment variable sets the shell used to run operating system commands. The default shell is the Bourne shell. If the shell cannot be run, then SQL*Plus displays an error message.

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

For example, to run a single command, use the following command syntax:

```
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, run the HOST or ! command and then press **Enter**. You are returned 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 name 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 the following SQL*Plus restrictions:

- Resizing Windows
- Return Codes
- Hiding Your Password

Resizing Windows

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

Return Codes

Operating system 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, then the output of the `ps` command may 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, then 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, then you should consider using operating system pipes in script files to pass a password to SQL*Plus. For example:

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

Alternatively, run the following commands:

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

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

5

Configuring Oracle Net Services

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

- Locating 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:** *Oracle Database Net Services Administrator's Guide* for more information about Oracle Net Services

Locating 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 type of 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 this environment variable is 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 this environment variable is set
2. One of the following directories:
   - On Solaris:

     `/var/opt/oracle`

   - On other platforms:

     `/etc`
3. The `$ORACLE_HOME/network/admin` directory

For some system-level configuration files, users may 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 select 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, run the following commands:

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

On an Oracle Database Client system, 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
```

See Also: *Oracle Database Net Services Administrator's Guide* for more information about the `adapters` utility

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 Secure Sockets Layer (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.

See Also:

- 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, refer to *Oracle Database Advanced Security Administrator's Guide*
- *Oracle Database Net Services Administrator's Guide* for more information about Oracle protocol support

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 that is different from any other name used for an IPC KEY on the same system.

The following is 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 Oracle Database, whether they are installed on the same or different systems. If the TCP/IP protocol is installed on your system, then 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 is 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 Database instances through TCP/IP and SSL. To use TCP/IP with SSL, you must install Oracle Advanced Security.

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 is 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, then 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 how to use Oracle precompilers and the Oracle Call Interface. It contains the following sections:

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

Overview of Oracle Precompilers

Oracle precompilers are application development tools that are 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.

It contains the following sections:

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++	`pcscfg.cfg`
Pro*COBOL (AIX, HP-UX, Solaris, Tru64 UNIX, and zSeries Linux)	`pcbcfg.cfg`
Pro*FORTRAN (AIX, HP-UX, Solaris, and Tru64 UNIX)	`pccfor.cfg`
Object Type Translator	`ottcfg.cfg`
SQL*Module for Ada (AIX)	`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 relink exename = executable_name
```

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, replace *`executable`* with one of the product executables listed in Table 6–2.

Table 6–2 Executables for Oracle Precompilers

Product	Executable
Pro*C/C++	`proc`
Pro*COBOL (AIX, HP-UX, Tru64 UNIX, Solaris SPARC (64-bit) and zSeries Linux)	`procob` or `rtsora`
Pro*COBOL 32-bit (AIX, HP-UX PA-RISC, Solaris SPARC (64-bit), Solaris x86, and zSeries Linux)	`procob32` or `rtsora32`
Pro*FORTRAN (AIX, HP-UX, Solaris, and Tru64 UNIX)	`profor` (32-bit for AIX and Solaris)
Pro*FORTRAN 32-bit (HP-UX)	profor32
SQL*Module for Ada (AIX)	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++	`$ORACLE_HOME/precomp/doc/proc2/readme.doc`
Pro*COBOL	`$ORACLE_HOME/precomp/doc/procob2/readme.doc`
Pro*FORTRAN	`$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, then verify that the case of 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 performs the same way when it is run without the debugger.

- Value of `IRECLEN` and `ORECLEN` parameters

  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 run time. Libraries that are linked at run time are called dynamic or shared libraries. The benefits of dynamic linking are:

- Reduced disk space requirements: More than one application or call to the same application can use the same dynamic library.
- Reduced 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 `$ORACLE_HOME/lib` or `$ORACLE_HOME/lib32` directories. If you use the Oracle-provided `demo_`*`product`*`.mk` make file to link an application, then the client shared library is linked by default.

If the shared library path environment variable setting does not include the directory that contains the client shared library, then you may see an error message similar to one of the following lines when starting an executable:

```
Cannot load library libclntsh.a
Can't open shared library: .../libclntsh.sl.10.1
```

```
libclntsh.so.10.1: can't open file: errno=2
can't open library: .../libclntsh.dylib.10.1
Cannot map libclntsh.so
```

To avoid this error, set the shared library path environment variable to specify the appropriate directory. The following table shows sample settings for this environment variable name. If your platform supports both 32-bit and 64-bit applications, then ensure that you specify the correct directory, depending on the application that you want to run.

Platform	Environment Variable	Sample Setting
AIX (32-bit applications)	LIBPATH	$ORACLE_HOME/lib32
AIX (64-bit applications)	LIBPATH	$ORACLE_HOME/lib
HP-UX (32-bit applications)	SHLIB_PATH	$ORACLE_HOME/lib32
HP-UX (64-bit applications), Linux, and Tru64 UNIX	LD_LIBRARY_PATH	$ORACLE_HOME/lib
Mac OS X	DYLD_LIBRARY_PATH	$ORACLE_HOME/lib
zSeries Linux (31-bit applications) and Solaris (32-bit applications)	LD_LIBRARY_PATH	$ORACLE_HOME/lib32
Solaris (32-bit applications)	LD_LIBRARY_PATH_64	$ORACLE_HOME/lib

The client shared library is created automatically during installation. If you have to re-create it, then:

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

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

Nonthreaded Client Shared Library

> **Note:** The information in this section applies to HP-UX PA-RISC systems.

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, run 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
  ```

Bit-Length Support for Client Applications

The following table identifies the bit lengths (31-bit, 32-bit, or 64-bit) supported for client applications.

Client Application Type	Supported Platforms
32-bit only	Mac OS X, Linux x86, and Solaris x86
64-bit only	Tru64 UNIX and Linux Itanium
32-bit and 64-bit	Solaris SPARC and Linux x86-64
31-bit and 64-bit	zSeries Linux

On AIX, HP-UX, Solaris SPARC, and zSeries Linux, all demonstrations and client applications provided with Oracle Database 10g release 2 (10.2) link and run in 64-bit mode. On AIX, Solaris SPARC, and HP-UX, you can build 32-bit and 64-bit client applications in the same Oracle home directory. Similarly, on zSeries Linux, you can build 31-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.

Platform	32-Bit (or 31-Bit) Client Shared Library	64-Bit Client Shared Library
AIX	`$ORACLE_HOME/lib32/libclntsh.a` `$ORACLE_HOME/lib32/libclntsh.so`	`$ORACLE_HOME/lib/libclntsh.a` `$ORACLE_HOME/lib/libclntsh.so`
HP-UX PA-RISC	`$ORACLE_HOME/lib32/libclntsh.sl`	`$ORACLE_HOME/lib/libclntsh.sl`
HP-UX Itanium, Solaris, Linux x86-64, and zSeries Linux	`$ORACLE_HOME/lib32/libclntsh.so` **Note:** On zSeries Linux, the `$ORACLE_HOME/lib32` directory contains 31-bit libraries.	`$ORACLE_HOME/lib/libclntsh.so`

To implement a mixed word-size installation:

1. Run the following command to generate the 32-bit and 64-bit client shared libraries:

   ```
   $ $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, Linux x86, Linux x86-64, Solaris, and Tru64 UNIX	`LD_LIBRARY_PATH`

Building 32-Bit Pro*C and OCI Customer Applications

If the operating system supports both 32-bit and 64-bit Pro*C and Oracle Call Interface (OCI) customer applications, then you can find more information about building 32-bit Pro*C and OCI applications in the following files:

For information about. . .	Refer to 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`

32-Bit Executables and Libraries

Note: Information in this section applies to the AIX, HP-UX, Solaris SPARC, and zSeries Linux platforms.

If the platform supports both 32-bit and 64-bit applications, then the `$ORACLE_HOME/bin` directory contains both 32-bit and 64-bit executables. In addition, the following directories contain 32-bit libraries:

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

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:

- *Oracle Database Installation Guide* for information about supported compiler versions
- *Pro*C/C++ Programmer's Guide* for information about the Pro*C/C++ precompiler and interface features

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 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 demonstration programs, 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, which is 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, run the following command:

> **Note:** On AIX systems, to ensure that the demonstration programs compile correctly, include the -r option of the make command in the following examples. For example:
>
> ```
> $ make -r -f demo_proc.mk sample1
> ```

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

To create all the C demonstration programs for Pro*C/C++, run the following command:

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

To create all the C++ demonstration programs for Pro*C/C++, run the following command:

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

To create all the Object demonstration programs for Pro*C/C++, run the following command:

```
$ 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, then a message prompting you to run it is displayed.

To build a demonstration program and run the corresponding SQL script, include the make macro argument RUNSQL=run at the command line. For example, to create the sample9 demonstration program and run the required $ORACLE_HOME/precomp/demo/sql/sample9.sql script, run the following command:

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

To create all the Object demonstration programs and run all the required SQL scripts, run the following command:

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

Pro*C/C++ User Programs

You can use the $ORACLE_HOME/precomp/demo/proc/demo_proc.mk make file to create user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the demo_proc32.mk make file to build 32-bit (or 31-bit on zSeries Linux) user programs. The following table shows the make files that you can use to build 32-bit (or 31-bit) and 64-bit user programs with Pro*C/C++.

Platform	64-bit Make File	32-Bit Make File
AIX, HP-UX, Solaris SPARC, and zSeries Linux	demo_proc.mk	demo_proc32.mk **Note:** On zSeries Linux, this make file builds 31-bit user programs.
Linux x86, Mac OS X, and Solaris	NA	demo_proc.mk

Platform	64-bit Make File	32-Bit Make File
Linux Itanium and Tru64 UNIX	demo_proc.mk	NA

See Also: The make file for more information about creating user programs

Note: On AIX systems, to ensure that your programs compile correctly, specify the `-r` option for the `make` command used in the following examples.

To create a program by using the `demo_proc.mk` make file, run 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`, run 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, run the following command:

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

- For C source statically linked with the client shared library, run the following command:

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

- For C++ source dynamically linked with the client shared library, run the following command:

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

- For C++ source statically linked with the client shared library, run the following command:

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

Pro*COBOL Precompiler

Note: Pro*COBOL is not supported on Linux Itanium, Solaris x86, and Mac OS X.

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

*Table 6–4 Pro*COBOL Naming Conventions*

Item	Naming Convention
Executable	`procob` or `procob32`
Demonstration directory	`procob2`
Make file	`demo_procob.mk` or `demo_procob_32.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.

Acucorp ACUCOBOL-GT COBOL Compiler

To use the Acucorp ACUCOBOL-GT COBOL compiler, you must set the `A_TERMCAP`, `A_TERM`, `PATH`, and `LD_LIBRARY_PATH` environment variables. If the `LD_LIBRARY_PATH` environment variable setting does not include the correct directory, then an error message similar to the following is displayed when you compile or run a program:

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

A_TERMCAP and A_TERM

Set the `A_TERMCAP` environment variable to specify the location of the `a_termcap` file and set the `A_TERM` environment variable to specify a supported terminal from that file. For example:

- Bourne, Bash, or Korn shell:

```
$ A_TERMCAP=/opt/COBOL/etc/a_termcap
$ A_TERM=vt100
$ export A_TERMCAP A_TERM
```

- C shell:

```
% setenv A_TERMCAP /opt/COBOL/etc/a_termcap
% setenv A_TERM vt100
```

PATH

Set the `PATH` environment variable to include the `/opt/COBOL/bin` directory (or the `/opt/COBOL31/bin` directory if you are using the 31-bit compiler on zSeries Linux systems):

- Bourne, Bash, or Korn shell:

```
$ PATH=/opt/COBOL/bin:$PATH
$ export PATH
```

- C shell:

```
% setenv PATH opt/COBOL/bin:${PATH}
```

LD_LIBRARY_PATH

> **Note:** On AIX, the LIBPATH variable is the LD_LIBRARY_PATH variable equivalent. You must use the LIBPATH variable on AIX instead of the LD_LIBRARY_PATH variable in the following commands.

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 /opt/COBOL/lib directory, then run the following command:

- Bourne, Bash, or Korn shell:

```
$ LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/COBOL/lib
$ export LD_LIBRARY_PATH
```

- C shell:

```
% setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/opt/COBOL/lib
```

Micro Focus Server Express COBOL Compiler

To use the Micro Focus Server Express COBOL compiler, you must set the COBDIR and PATH environment variables and the shared library path environment variable.

> **See Also:** The "Client Shared and Static Libraries" section on page 6-3 for information about the shared library path environment variable

If the shared library path environment variable setting does not include the $COBDIR/coblib directory, then an error message similar to the following is displayed when you compile or run a program:

- On Linux:

```
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 HP-UX PA-RISC and Solaris SPARC:

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

- On AIX:

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

- On HP-UX Itanium:

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

COBDIR

Set the COBDIR environment variable to the directory where the compiler is installed. For example, if the compiler is installed in the /opt/lib/cobol directory, then run the following command:

- Bourne, Bash, or Korn shell:

```
$ COBDIR=/opt/lib/cobol
$ export COBDIR
```

- C shell:

```
% setenv COBDIR /opt/lib/cobol
```

PATH

Set the PATH environment variable to include the $COBDIR/bin directory:

- Bourne, Bash, or Korn shell:

```
$ PATH=$COBDIR/bin:$PATH
$ export PATH
```

- C shell:

```
% setenv PATH ${COBDIR}/bin:${PATH}
```

Shared Library Path

Set the LIBPATH, LD_LIBRARY_PATH, or SHLIB_PATH environment variable to the directory where the compiler library is installed. For example, if the platform uses the LD_LIBRARY_PATH environment variable and the compiler library is installed in the $COBDIR/coblib directory, then run the following command:

- Bourne, Bash, or Korn shell:

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

- C shell:

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

Pro*COBOL Oracle Runtime System

Oracle provides its own complete run-time 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) run-time system instead of the cobrun run-time system to run dynamically loadable Pro*COBOL programs. If you attempt to run a Pro*COBOL program with cobrun, then an error message similar to the following is displayed:

```
$ 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 demonstration programs, 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 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, run the following command:

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

To create the Pro*COBOL demonstration programs, run the following command:

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

To create and run a dynamically loadable `sample1.gnt` program to be used with the `rtsora` run-time system, run the following command:

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

Some demonstration programs require you to run a SQL script, which is located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, then a message requesting you to run it is displayed.

To build a demonstration program and run the corresponding SQL script, include the `make` macro argument `RUNSQL=run` in the command. For example, to create the `sample9` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/sample9.sql` script, run the following command:

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

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

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

Pro*COBOL User Programs

You can use the `$ORACLE_HOME/precomp/demo/procob2/demo_procob.mk` make file to create user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the `demo_procob_32.mk` make file to build 32-bit (or 31-bit) user programs. The following table shows the make files that you can use to build 32-bit (or 31-bit) and 64-bit user programs with Pro*COBOL.

Platform	64-bit Make File	32-Bit Make File
AIX, HP-UX, Solaris SPARC, and zSeries Linux	`demo_procob.mk`	`demo_procob_32.mk` **Note:** On zSeries Linux, this make file builds 31-bit user programs.
Linux x86	Not applicable	`demo_procob.mk`
Tru64 UNIX	`demo_procob.mk`	Not applicable

See Also: The make file for more information about creating user programs

To create a program using the `demo_procob.mk` make file, run 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`, run 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, run the following command:

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

- For COBOL source, statically linked, run the following command:

  ```
  $ 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), run the following command:

  ```
  $ 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`, then 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`, then columns 1 to 6 are dropped, making column 7 the leftmost column.

Pro*FORTRAN Precompiler

Note: Pro*FORTRAN is not supported on Linux or Mac OS X.

Before you use the Pro*FORTRAN precompiler, verify that the correct version of the compiler is installed. This section contains the following topics:

- Pro*FORTRAN Demonstration Programs
- Pro*FORTRAN User Programs

See Also:

- *Oracle Database Installation Guide for Linux x86* for information about supported compiler versions
- *Pro*FORTRAN Supplement to the Oracle Precompilers Guide* for information about the Pro*FORTRAN precompiler and interface features

Pro*FORTRAN Demonstration Programs

Demonstration programs are provided to show the features of the Pro*FORTRAN precompiler. All 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 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.

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

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

To create the Pro*FORTRAN demonstration programs, run the following command:

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

Some demonstration programs require you to run a SQL script that is located in the `$ORACLE_HOME/precomp/demo/sql` directory. If you do not run the script, then a message prompting you to run it is displayed.

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, run the following command:

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

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

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

Pro*FORTRAN User Programs

You can use the `$ORACLE_HOME/precomp/demo/profor/demo_profor.mk` make file to create user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the `demo_profor_32.mk` make file to build 32-bit user programs. The following table shows the make files that you can use to build 32-bit and 64-bit user programs with Pro*FORTRAN.

Platform	64-bit Make File	32-Bit Make File
AIX, HP-UX, and Solaris SPARC	`demo_procob.mk`	`demo_procob_32.mk`
Tru64 UNIX	`demo_procob.mk`	NA

> **See Also:** The make file for more information about creating user programs

To create a program using the demo_proc.mk make file, run 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, run 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, run the following command:

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

- For an executable statically linked with the client shared library, run the following command:

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

SQL*Module for ADA

Note: The information in this section applies to the AIX platform.

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

See Also:

- *Oracle Database Installation Guide for UNIX Systems* for information about required compiler versions
- *Oracle SQL*Module for Ada Programmer's Guide* for information about SQL*Module for Ada

SQL*Module for Ada Demonstration Programs

Demonstration programs are provided to show the features of SQL*Module for Ada. All 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.

Run the following command to create the SQL*Module for Ada demonstration programs, run the necessary SQL scripts to create the MODTEST user, and create the sample college database:

```
$ 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, run the following command:

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

To create the SQL*Module for Ada demonstration programs, without re-creating the sample college database, run the following command:

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

To create a single demonstration program (demohost), without re-creating the sample college database, run the following command:

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

To run the programs, you must define an Oracle Net connect string or alias named INST1_ALIAS that 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, run 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:** *Oracle SQL*Module for Ada Programmer's Guide* for information about SQL*Module for Ada

OCI and OCCI

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:

- *Oracle Database Installation Guide for Linux x86* for information about supported compiler versions
- *Oracle Call Interface Programmer's Guide* or *Oracle C++ Call Interface Programmer's Guide* for information about OCI and OCCI

OCI and OCCI Demonstration Programs

Demonstration programs that show the features of OCI and OCCI are provided with the software. There are two types of demonstration programs: C and C++. All 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 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_rdbms.mk` make file, which is located in the `$ORACLE_HOME/rdbms/demo` directory, to create the demonstration programs. For example, to compile and link the `cdemo1` demonstration program, run the following command:

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

To create the C demonstration programs for OCI, run the following command:

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

To create the C++ demonstration programs for OCCI, run the following command:

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

OCI and OCCI User Programs

You can use the `$ORACLE_HOME/rdbms/demo/demo_rdbms.mk` make file to create user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the `demo_rdbms32.mk` make file to build 32-bit user programs. The following table shows the make files that you can use to build 32-bit and 64-bit user programs with Pro*FORTRAN.

Platform	64-bit Make File	32-Bit Make File
AIX, HP-UX, Solaris SPARC, and zSeries Linux	`demo_rdbms.mk`	`demo_rdbms32.mk` **Note:** On zSeries Linux, this make file builds 31-bit user programs.
Linux x86, Mac OS X, and Solaris x86	NA	`demo_rdbms.mk`
Linux Itanium and Tru64 UNIX	`demo_rdbms.mk`	NA

See Also: The make file for more information about creating user programs

To create a program by using the `demo_rdbms.mk` make file, run a command similar to the following:

```
$ 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`, run 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, run the following command:

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

- For C source, statically linked, run the following command:

  ```
  $ 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`, run 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, run the following command:

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

- For C++ source, statically linked, run the following command:

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

Oracle JDBC/OCI Programs with a 64-Bit Driver

Note:

- The information in this section applies to AIX, HP-UX, Solaris SPARC, and zSeries Linux platforms.
- You can use the instructions and make files described in this section to create JDBC/OCI user programs that use a 64-bit driver.

To run JDBC/OCI demonstration programs with a 64-bit driver:

1. Add `$ORACLE_HOME/jdbc/lib/ojdbc14.jar` to the start of the `CLASSPATH` environment variable value 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 and the -d64 flag as follows:

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

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 LD_LIBRARY_PATH_64 environment variable to include the $ORACLE_HOME/lib directory.

Note: On AIX, the LIBPATH variable is the LD_LIBRARY_PATH_64 variable equivalent. You must use the LIBPATH variable on AIX instead of the LD_LIBRARY_PATH_64 variable.

Custom Make Files

Oracle recommends that you use the demo_*product*.mk make files provided with the software 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, remember that 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 the symbols are resolved during linking.

  Except for AIX, the order of the Oracle libraries is essential on all platforms for the following reasons:

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

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

- If you want to add your library to the link line, then 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, then you must be aware of how macro and suffix processing differs from the make utility provided with the operating system. Oracle make files are tested and supported with the make utility.
- 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.

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 is displayed, then verify 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, which is used to locate the `sqlcex` symbol:

```
$ symfind sqlcex

SymFind - Find Symbol <sqlcex> in <**>.a, .o, .so
-----------------------------------------------------
Command:         /u01/app/oracle/product/10.2.0/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/10.2.0
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
```

Multithreaded Applications

The Oracle libraries provided with this release are thread-safe. This enables support for multithreaded applications.

Using Signal Handlers

Oracle Database uses signals for two-task communication. Signals are installed in a user process when the process 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 operating system kernel sends a SIGCLD signal to the user process. The signal handler uses the `wait()` routine to determine if 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, then 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.

Table 6–5 (Cont.) Signals for Two-Task Communication

Signal	Description
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, then 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, then 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 the user interrupt handler */
        if (err = osnsui(&handle, sig_handler, (char *) 0))
        {
                /* If the return value is nonzero, then an error has occurred
                   Take appropriate action for the error. */
                ...
        }

        ...

        /* Clear the interrupt handler */
        if (err = osncui(handle))
        {
                /* If the return value is nonzero, then 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 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 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`, `libclntsh.so`, or libclntsh.dylib depending on your platform) and the client static library (`libclntst10.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. 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 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. 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. Run 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 with the software:

- `examp1.sql` to `examp8.sql`
- `examp11.sql` to `examp14.sql`
- `sample1.sql` to `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. Run 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:

1. If required, 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 required, add an entry for external procedures to the `listener.ora` file, similar to the following:

> **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.

   - On 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)
      )
    )
```

   - On 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)
    )
  )
```

- On Mac OS X:

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

3. Change directory to `$ORACLE_HOME/plsql/demo`.
4. Run the following command to create the `extproc.so` shared library, build the required database objects, and load the sample data:

> **Note:** On Mac OS X systems, use the shared library name `extproc.dylib` in the following examples.

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

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

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

5. From SQL*Plus, run 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 start the demonstration, run 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 the PL/SQL precompiler demonstrations, set the library path environment variable to include the $ORACLE_HOME/lib directory, and run the following commands:

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

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

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

To start the examp9 demonstration, run the following command:

```
$ ./examp9
```

Calling 32-Bit External Procedures from PL/SQL

Note: This section applies to AIX, HP-UX, Solaris SPARC, and zSeries Linux.

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, Solaris SPARC, and zSeries Linux systems. To enable 32-bit external procedures:

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

   ```
   (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
zSeries Linux and 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 have to support both 32-bit and 64-bit external procedures.

8

Tuning Oracle Database

This chapter describes how to tune Oracle Database. 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.

Before tuning the database, you must observe its normal behavior by using the tools described in the "Operating System Tools" section on page 8-1.

Operating System Tools

Several operating system tools are available to enable you to 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.

This section provides information about the following common tools:

See Also: The operating system documentation and man pages for more information about these tools

vmstat

Note: On Mac OS X, the `vm_stat` command displays virtual memory information. Refer to the `vm_stat` man page for more information about using this command.

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

- On HP-UX and Solaris:

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

- AIX, Linux, and Tru64 UNIX:

```
$ vmstat 5 6
```

The following is sample output of this command on HP-UX:

```
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` sub 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, then swapping occurs and the 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 sufficient available memory.

Note: The output from the `vmstat` command differs across platforms.

See Also: Refer to the man page for information about interpreting the output

sar

Depending on the switches that you supply with the command, use the `sar` (system activity reporter) command to display cumulative activity counters in the operating system.

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

On an HP-UX system, the following command displays a summary of I/O activity ten times, at ten-second intervals:

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

The following example shows the output of this command:

```
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
```

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

> **See Also:** The man page for more information about `sar`

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 can be used to balance I/O loads.

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

```
$ iostat 5 5
```

The following is sample output of 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

> **See Also:** The Determining Available and Used Swap Space section on page D-1 for information about swap space on Mac OS X systems

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 and Tru64 UNIX	`swapon -s`
Solaris	`swap -l` and `swap -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.

- Base Operation System Tools
- Performance Toolbox
- System Management Interface Tool

> **See Also:** The AIX operating system documentation and man pages for more information about these tools

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	Function
`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

Tool	Function
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

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 monitoring and tuning tools listed in the following table:

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 blocks of a file 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:

- *Performance Toolbox for AIX Guide and Reference* for information about these tools
- *AIX 5L Performance Management Guide* for information about the syntax of some of these tools

System Management Interface Tool

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

HP-UX Tools

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

- GlancePlus/UX

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

- HP PAK

  HP Programmer's Analysis Kit (HP PAK) consists of the following tools:

  – Puma

    This tool collects performance statistics during a program run. It provides several graphical displays for viewing and analyzing the collected statistics.

  – Thread Trace Visualizer (TTV)

    This tool displays trace files produced by the instrumented thread library, `libpthread_tr.sl`, in a graphical format. It enables 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.

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

Tools	Function
`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.

Tools	Function
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.

Linux Tools

On Linux systems, use the `top`, `free`, and `cat /proc/meminfo` commands 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 restart. 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
```

Mac OS X Tools

You can use the following additional performance tuning tools:

- Use the `top` command to display information about running processes and memory usage.
- Use the Apple Computer Hardware Understanding Developer (CHUD) tools, such as Shark and BigTop, to monitor system activity and tune applications.

See Also: For more information about the CHUD tools, refer to

http://developer.apple.com/documentation/Performance/Conceptual/PerformanceFundamentals/index.html

Tuning Memory Management

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

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

This section includes the following topics:

- Allocating Sufficient Swap Space
- Controlling Paging
- Adjusting Oracle Block Size

Allocating Sufficient Swap Space

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

If your system is swapping and you must conserve memory, then:

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

> **Note:** On Mac OS X systems, swap space is allocated dynamically. If the operating system requires more swap space, then it creates additional swap files in the `/private/var/vm` directory. Ensure that the file system that contains this directory has sufficient free disk space to accommodate additional swap files.

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

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

Platform	Command
Tru64 UNIX	`swapon -a`

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

> **See Also:** The operating system documentation for more information about these commands

Controlling Paging

Paging may 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 may 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` (`vm_stat` on Mac OS X) or `sar` command to monitor paging.

> **See Also:** The man pages or your operating system documentation for information about interpreting the results for your platform

The following table lists the important columns from the output of these commands.

Platform	Column	Function
Solaris	`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.
Solaris	`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.
HP-UX	`at`	Indicates the number of address translation page faults. Address translation faults occur when a process refers to a valid page not in memory.
HP-UX	`re`	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, then consider the following solutions:

- Install more memory.
- Move some of the work to another system.
- Configure the System Global Area (SGA) to use less memory.

Adjusting Oracle Block Size

During read operations, entire operating system blocks are read from the disk. If the database block size is smaller than the operating system file system block size, then I/O bandwidth is inefficient. If you set Oracle Database block size to be a multiple of the file system block size, then you can increase performance by up to 5 percent.

The `DB_BLOCK_SIZE` initialization parameter sets the database block size. However, to change the value of this parameter, you must re-create the database.

To see the current value of the DB_BLOCK_SIZE parameter, run 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.

Using Automatic Storage Management

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

By using Automatic Storage Management, you avoid manually tuning disk I/O.

Choosing 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. This fact can have a substantial impact on database performance. The following table lists common file system types.

File System	Platform	Description
S5	HP-UX and Solaris	UNIX System V file system
UFS	AIX, HP-UX, Mac OS X, Solaris, Tru64 UNIX	Unified file system, derived from BSD UNIX **Note:** On Mac OS X, Oracle does not recommend the use of the UFS file system for either software or database files.
VxFS	AIX, HP-UX, and Solaris	VERITAS file system
None	All	Raw devices (no file system)
ext2/ext3	Linux	Extended file system for Linux
OCFS	Linux	Oracle cluster file system
AdvFS	Tru64 UNIX	Advanced file system
CFS	Tru64 UNIX	Cluster file system
JFS/JFS2	AIX	Journaled file system
HFS Plus, HFSX	Mac OS X	HFS Plus is the standard hierarchical file system used by Mac OS X. HFSX is an extension to HFS Plus that enables case-sensitive file names.
GPFS	AIX	General parallel file system

The suitability of a file system for 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, then:

- 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, then consider placing log files on a different file system from data files.

Monitoring Disk Performance

The following sections describe the procedure for monitoring disk performance.

Monitoring Disk Performance on Mac OS X

Use the `iostat` and `sar` commands to monitor disk performance. For more information about using these commands, refer to the man pages.

Monitoring Disk Performance on Other Operating Systems

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

The following table describes the columns of the `sar -b` command output that are significant for analyzing disk performance.

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 spent 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` column values 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 2 drives and no more than 60 for 4 to 8 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, then the system is I/O bound.

System Global Area

The 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` (`shm_max` on Tru64 UNIX) kernel parameter.

The following table shows the recommended value for this parameter, depending on your platform:

Platform	Recommended Value
AIX	NA
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-1 for information about the `shmmax` parameter on HP-UX.
Linux	Half the size of the physical memory installed on the system
Mac OS X	NA
	The largest SGA size on Mac OS X is 1 MB.
Solaris and 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 Database instance to start. If your system runs both Oracle9*i* and Oracle Database 10*g* instances, then you must set the value of this parameter to 2 GB minus 16 MB. On Solaris, this value can be greater than 4 GB on 64-bit systems.

If the size of the SGA exceeds the maximum size of a shared memory segment (`shmmax` or `shm_max`), then Oracle Database 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 automatic tuning of the SGA size.

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

An Oracle Database configured with Shared Server requires 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 Oracle Universal Installer, then the value of the `SHARED_POOL_SIZE` parameter is set automatically by Oracle Database Configuration Assistant. However, if you created a

database manually, then increase the value of the SHARED_POOL_SIZE parameter in the parameter file by 1 KB for each concurrent user.

Determining the Size of the SGA

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

- Run 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.

- When you start your database instance, the size of the SGA is displayed next to the Total System Global Area heading.
- On systems other than Mac OS X, run the ipcs command as the oracle user.

Shared Memory on AIX

Note: The information in this section applies only to 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.

Shared memory can be pinned to prevent paging and to reduce I/O overhead. To perform 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.

Run the following command to make pinned memory available to Oracle Database:

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

Run 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, enabling 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 run 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- and POWER5-Based Systems

To turn on and reserve 10 large pages each of size 16 MB on a POWER4 or POWER 5 system, run the following command:

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

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

Oracle recommends specifying enough large pages to contain the entire SGA. The Oracle Database 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, then the portion of the SGA exceeding these sizes is allocated to ordinary shared memory.

> **See Also:** The AIX documentation for more information about enabling and tuning pinned memory and large pages

Tuning the Operating System Buffer Cache

To take full advantage of raw devices, adjust the size of Oracle Database buffer cache. If memory is limited, then adjust 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.

Oracle Database buffer cache is the area in memory that stores Oracle Database buffers. Because Oracle Database can use raw devices, it does not use the operating system buffer cache.

If you use raw devices, then increase the size of Oracle Database buffer cache. If the amount of memory on the system is limited, then 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.

> **See Also:** The man page on Tru64 UNIX for more information about the `sar` command

---

Note: On 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 includes the following topics:

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.

This section contains the following topics:

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 may 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.

It is possible to tune buffer-cache paging activity, but you must do it carefully and infrequently. Use the `/usr/sbin/vmo` command to tune the AIX system parameters in the following table.

Parameter	Description
minfree	The minimum free-list size. If the free-list space in the buffer falls below this size, then 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, then the system stops using page stealing to replenish the free list.
minperm	The minimum number of permanent buffer pages for file I/O.
maxperm	The maximum number of permanent buffer pages for file I/O.

See Also: *AIX 5L Performance Management Guide* for more information about AIX system parameters

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, then disk usage increases and potentially saturates one or more disks. If the cache is too large, then memory is wasted.

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, then increasing the minperm parameter value may help. If maintaining a high buffer hit ratio is not critical, then 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 values according to the following guidelines:

- If the percentage of real memory occupied by file pages falls below the minperm value, then the virtual memory manager (VMM) page-replacement algorithm takes both file and computational pages, regardless of repage rates.
- If the percentage of real memory occupied by file pages rises above the maxperm value, then the VMM page-replacement algorithm takes both file and computational pages.
- If the percentage of real memory occupied by file pages is between the minperm and maxperm parameter values, then the VMM normally takes only file pages. However, if the repaging rate for file pages is higher than the repaging rate for computational pages, then the computational pages are taken as well.

Use the following algorithm to calculate the default values:

- minperm (in pages) = ((number of page frames)-1024) * 0.2
- maxperm (in pages) = ((number of page frames)-1024) * 0.8

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:

```
à# /usr/sbin/vmo -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, then 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 may be better used for other purposes, such as for the Oracle System Global Area.

Allocating Sufficient Paging Space

Inadequate paging space (swap space) usually causes the system to stop responding or show very 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.

If paging space is preallocated, then 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 required. The system uses swap space only if it runs out of real memory. If the memory is sized correctly, then 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 percent on a healthy system. A properly sized deployment requires very little paging space and an excessive amount of swapping is an indication that the RAM on the system may be undersized.

Caution: Do not undersize the paging space. If you do, then the system terminates active processes when it runs out of space. However, oversizing 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 Database 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, then 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 may look like the following:
  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 example, suppose that you query Oracle Database dynamic performance tables and views and find that both the shared pool and database buffer cache require more memory. Then, assigning the limited spare memory to the shared pool may 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 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 Oracle Database is installed on a journaled file system, then the block size should be a multiple of the file system block size (4 KB on JFS, 16 K to 1 MB on GPFS). For databases on raw partitions, 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 may 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* Database.

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 maximize CPU usage, and by doing this, increase overall throughput.

The number of buffers (set by the SQL*Loader `BUFFERS` parameter) you choose depends on the amount of available memory and how much you want to maximize CPU usage.

The performance gains depend on CPU usage and the degree of parallelism that you use when loading data.

> **See Also:** *Oracle Database Utilities* for information about adjusting the file processing options string for the `BUFFERS` parameter and for information about the SQL*Loader utility

BUFFER Parameter for the Import Utility

The `BUFFER` parameter for the Import utility should be set to a large value to optimize the performance of high-speed networks when they are used. For example, if you use the IBM RS/6000 Scalable POWERparallel Systems (SP) switch, then 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.

Ensure 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 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 enables you to spread I/O more evenly across disks, resulting in greater overall performance.

> **Note:** Do not add logical volumes to Automatic Storage Management disk groups. Automatic Storage Management works best when you add raw disk devices to disk groups. If you are using Automatic Storage Management, then 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 items listed in the following table.

Item	Recommended Settings
Drives	There must be at least two physical drives. The drives should have minimal activity when performance-critical sequential I/O is carried out. Sometimes, you must stripe the logical volume between two or more adapters.
Stripe unit size	Although the stripe unit size can be any power of 2 (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.

Item	Recommended Settings
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 expect significant performance gains.

Using Journaled File Systems Compared to Raw Logical Volumes

Address the following considerations when deciding whether or not 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 the kernel.
- Different vendors implement the file system layer in different ways to capitalize on 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 features included in AIX 5L improve file system performance to a level comparable to raw logical volumes.

If you use a journaled file system, then 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 RAC 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 enables 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 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.
JFS2	cio	Concurrent I/O is a better setting than Direct I/O on JFS2, because it provides support for multiple concurrent readers and writers on the same file. However, due to AIX restrictions on JFS2/CIO, Concurrent I/O is intended to be used only with Oracle data files, control files, and log files. It should be applied only to file systems that are dedicated to such a purpose. For the same reason, the Oracle home directory is not supported on a JFS2 file system mounted with the `cio` option. For example, during installation, if you inadvertently specify that the Oracle home directory is on a JFS2 file system mounted with the CIO option, then while trying to relink Oracle, you may encounter the following error: `"ld: 0711-866 INTERNAL ERROR: Output symbol table size miscalculated"`
GPFS	NA	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

If you are placing Oracle Database logs on a JFS2 file system, then the optimal configuration is to create the file system using the `agblksize=512` option and to mount it with the `cio` option. This delivers logging performance within a few percentage points of the performance of a raw device.

Before Oracle Database 10*g*, Direct I/O and Concurrent I/O could not be enabled at the file level on JFS/JFS2. Therefore, the Oracle home directory and data files had to be placed in separate file systems for optimal performance. The Oracle home directory was placed on a file system mounted with default options, with the data files and logs on file systems mounted using the `dio` or `cio` options.

With Oracle Database 10*g*, you can enable Direct I/O and Concurrent I/O on JFS/JFS2 at the file level. You can do this by setting the `FILESYSTEMIO_OPTIONS` parameter in the server parameter file to `setall` or `directIO`. This enables Concurrent I/O on JFS2 and Direct I/O on JFS for all data file I/O. Because the `directIO` setting disables asynchronous I/O it should normally not be used. As a result of this 10g feature, you can place data files on the same JFS/JFS2 file system as the Oracle home directory and still use Direct I/O or Concurrent I/O for improved performance. As mentioned earlier, you should still place Oracle Database logs on a separate JFS2 file system for optimal performance.

Considerations for GPFS

If you are using GPFS, then you can use the same file system for all purposes. This includes using it for the Oracle home directory and for storing data files and logs. For optimal performance, you should 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 as long as 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 devices without having to manually reload all the data, perform the following steps as the `root` user:

1. Create a raw device (preferably, in a BigVG) using the new raw logical volume device type (`-T O`), which enables 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. In addition, you must bear in 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
   ```

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. However, 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 128 KB on AIX logical volumes or zero on AIX logical volumes created with the `mklv -T O` option.

When you have determined the offset, you can copy over 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=256
```

You can instruct Oracle Database to use a number of blocks smaller than the maximum capacity of a raw logical volume. If you do this, then you must add a `count` clause to ensure that only data that contains Oracle blocks is copied. 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 to the time it is completed. The number of `aioserver` servers determines the number of AIO requests that can be processed 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.

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:

- The System Management Interface Tool (SMIT) online Help for more information about SMIT
- The man pages for more information about the `smit aio` and `chdev` commands

Note: Starting with AIX 5L version 5.2, there are two AIO subsystems available. Oracle Database 10*g* uses Legacy AIO (`aio0`), even though the Oracle preinstallation script enables Legacy AIO (`aio0`) and POSIX AIO (`posix_aio0`). Both AIO subsystems have the same performance characteristics.

Set the minimum value to the number of servers to be started when the system is started. 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
`minservers`	Oracle recommends an initial value equal to the number of CPUs on the system or 10, whichever is lower.
`maxservers`	Starting with AIX 5L version 5.2, this parameter counts the maximum number of AIO servers per CPU. On previous versions of AIX, it was a systemwide value. If you are using GPFS, then set `maxservers` to worker1threads divided by the number of CPUs. This is the optimal setting. Increasing `maxservers` does not lead to improved I/O performance. If you are using JFS/JFS2, then set the initial value to 10 times the number of logical disks divided by the number of CPUs. 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.
`maxreqs`	Set the initial value to 4 times the number of logical disks multiplied by the queue depth. You can determine the queue depth by running the following command: `$ lsattr -E -l hdiskxx` Typically, the queue depth is 3.

If the value of the `maxservers` or `maxreqs` parameter is set too low, then the following warning messages are repeatedly displayed:

```
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 required. The changes take place when the system is restarted.

I/O Slaves

I/O Slaves are specialized Oracle processes that perform only I/O. They are rarely used on AIX, because 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 the 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
DBWR_IO_SLAVES	0 - 999	0
DB_WRITER_PROCESSES	1-20	1

Generally, you do not adjust the parameters in the preceding table. However, on large workloads, the database writer may become a bottleneck. If it does, then increase the value of `DB_WRITER_PROCESSES`. As a general rule, do not increase the number of database writer processes above one for each pair of CPUs in the system or partition.

There are times when you must 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` 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` parameter to greater than 0 only if the `DISK_ASYNCH_IO` or `TAPE_ASYNCH_IO` 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` parameter is 4.

Using the DB_FILE_MULTIBLOCK_READ_COUNT Parameter

When using Direct I/O or Concurrent I/O with Oracle Database 10*g*, the AIX file system does not perform any read-ahead on sequential scans. For this reason the `DB_FILE_MULTIBLOCK_READ_COUNT` value in the server parameter file should be increased when Direct I/O or Concurrent I/O is enabled on Oracle data files. 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, this

parameter ranges from 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 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. You can improve performance by doing this, 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:

```
à# /usr/sbin/vmo -o numclust=8
```

To disable write behind, enter the following command:

```
à# /usr/sbin/vmo -o numclust=0
```

Tuning Sequential Read Ahead

Note: The information in this section applies only to file systems, and only when neither Direct I/O nor Concurrent I/O are used.

The 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 consecutive pages of the file, the VMM assumes that the program continues 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 faster. The following VMM thresholds, implemented as kernel parameters, determine the number of pages it reads ahead:

- minpgahead

  This parameter stores the number of pages read ahead when the VMM first detects the sequential access pattern.

- maxpgahead

  This parameter stores the maximum number of pages that VMM reads ahead in a sequential file.

Set the minpgahead and maxpgahead parameters to appropriate values for an application. The default values are 2 and 8 respectively. Use the /usr/sbin/vmo 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, run the following command as the root user:

```
-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 enables 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.

Resilvering with Oracle Database

If you disable mirror write consistency (MWC) for an Oracle data file allocated on a raw logical volume (LV), then the Oracle Database crash recovery process uses resilvering to recover after a system failure. This resilvering process prevents database inconsistencies or corruption.

During crash recovery, if a data file is allocated on a logical volume with more than one copy, then the resilvering process performs a checksum on the data blocks of all the copies. It then performs one of the following:

- If the data blocks in a copy have valid checksums, then the resilvering process uses that copy to update the copies that have invalid checksums.
- If all copies have blocks with invalid checksums, then the resilvering process rebuilds the blocks using information from the redo log file. It then writes the data file to the logical volume and updates all the copies.

On AIX, the resilvering process works only for data files allocated on raw logical volumes for which MWC is disabled. Resilvering is not required for data files on mirrored logical volumes with MWC enabled, because MWC ensures that all copies are synchronized.

If the system crashes while you are upgrading an earlier release of Oracle Database that used data files on logical volumes for which MWC was disabled, then run the `syncvg` command to synchronize the mirrored LV before starting Oracle Database. If you do not synchronize the mirrored LV before starting the database, then Oracle Database may read incorrect data from an LV copy.

Note: If a disk drive fails, then resilvering does not occur. You must run the `syncvg` command before you can reactivate the LV.

Caution: Oracle supports resilvering for data files only. Do not disable MWC for redo log files.

Backing Up Raw Devices

Oracle recommends that you use RMAN to back up raw devices. If you do use the `dd` command to back up raw devices, then follow the instructions described in this section.

The offset of the first Oracle block on a raw device may be 0, 4K, or 128K depending on the device type. 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 `-T O` option. However, existing raw logical volumes created with earlier versions of Oracle Database typically have a nonzero offset. The following example shows how to back up and restore a raw device whose first Oracle block is at offset 4K:

```
$ 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
```

CPU Scheduling and Process Priorities

The CPU is another system component for which processes may 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), then there may be different levels of contention on each CPU.

The following sections provide information about CPU scheduling and process priorities:

- Changing Process Running Time Slice
- Using Processor Binding on SMP Systems

Changing Process Running Time Slice

The default value for the run-time slice of the AIX RR dispatcher is ten milliseconds (msec). Use the `schedo` command to change the time slice. A longer time slice causes a lower context switch rate if the average voluntary switch rate of the applications 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 run-time slice can deteriorate response time, especially on a uniprocessor system. The default run-time 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 may want to increase the time slice by entering the following command:

```
# /usr/sbin/schedo -t n
```

In the preceding command, setting *n* to 0 results in a slice of 10 msec, choosing a value of 1 results in a slice of 20 msec, choosing a value of 2 results in a slice of 30 msec, and so on.

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 enables Oracle Database processes to be scheduled optimally without processor binding. Therefore, Oracle no longer recommends binding processes to processors when running on AIX 5L version 5.2 or later.

Setting the AIXTHREAD_SCOPE Environment Variable

Threads in AIX can run with process-wide contention scope (M:N) or with systemwide contention scope (1:1). The `AIXTHREAD_SCOPE` environment variable controls which contention scope is used.

The default value of the `AIXTHREAD_SCOPE` environment variable is `P`, which specifies process-wide contention scope. When using process-wide contention scope, Oracle threads are mapped to a pool of kernel threads. When Oracle is waiting on an event and its thread is swapped out, it may return on a different kernel thread with a different thread ID. Oracle uses the thread ID to post waiting processes, so it is important for the thread ID to remain the same. When using systemwide contention scope, Oracle threads are mapped to kernel threads statically, one to one. For this reason, Oracle recommends that you use systemwide contention. The use of systemwide contention is especially critical for RAC instances.

In addition, on AIX 5L version 5.2 or later, if you set systemwide contention scope, then significantly less memory is allocated to each Oracle process.

Oracle recommends that you set the value of the `AIXTHREAD_SCOPE` environment variable to `S` in the environment script that you use to set the `ORACLE_HOME` or `ORACLE_SID` environment variables for an Oracle Database instance or an Oracle Net listener process as follows:

- Bourne, Bash, or Korn shell:

  Add the following line to the `~/.profile` or `/usr/local/bin/oraenv` script:

  ```
  AIXTHREAD_SCOPE=S; export AIXTHREAD_SCOPE
  ```

- C shell:

  Add the following line to the `~/.login` or `/usr/local/bin/coraenv` script:

  ```
  setenv AIXTHREAD_SCOPE S
  ```

Doing this enables systemwide thread scope for running all Oracle processes.

Network Information Service (NIS) external naming support

NIS external naming adapter is supported on AIX. To configure and use NIS external naming, refer to the "Configuring External Naming Methods" section of *Oracle Database Net Services Administrator's Guide*.

Simultaneous Multi-threading (SMT) on AIX 5.3

If Simultaneous Multi-threading (SMT) is enabled, and the AIX 5.3 operating system is being used, the `v$osstat` view reports 2 additional rows corresponding to the online logical(`NUM_LCPUS`) and virtual cpus (`NUM_VCPUS`).

If oracle is being run on AIX 5.2 or AIX 5.3 without SMT these rows are not reported.

B

Administering Oracle Database on HP-UX

This appendix provides information about administering Oracle Database on HP-UX. It contains the following topics:

- HP-UX Shared Memory Segments for an Oracle Instance
- HP-UX SCHED_NOAGE Scheduling Policy
- Lightweight Timer Implementation
- Asynchronous I/O
- Large Memory Allocations and Oracle Database Tuning
- CPU_COUNT Initialization Parameter and HP-UX Dynamic Processor Reconfiguration
- Network Information Service (NIS) external naming support

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, then 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 Oracle Database creates the instance. The number of protection keys available depends on the system architecture as shown in the following table.

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, then 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 requires only one protection key.

To display the list of active shared memory segments on the system, run the following command:

```
$ ipcs -m
```

If Oracle Database creates more segments for the instance than the number of protection keys, then increase the value of the `shmmax` kernel parameter.

> **See Also:** Oracle Database Installation Guide for UNIX Systems for more information about the recommended minimum kernel parameter values

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 does not provide the same level of performance gains in decision support environments because there is less resource competition. Because each application and server environment is different, you should test and verify that 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 stop responding.

Enabling SCHED_NOAGE for Oracle Database

To permit 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 restarts:

   ```
   dba RTPRIO RTSCHED
   ```

4. Save the file, and quit 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, then 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:** The HP-UX documentation, the rtsched(1) man page, and the rtsched(2) man page for more information about priority policies and priority ranges

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 enables 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 enables 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 permit Oracle Database to process 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, then add the MLOCK privilege on the same line.

```
dba RTPRIO RTSCHED MLOCK
```

4. Save the file, and quit 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, then you must use one of the following storage options for database files:

- Raw devices (partitions or logical volumes)
- An Automatic Storage Management disk group that uses raw partitions

> **See Also:** Oracle Database Installation Guide for UNIX Systems for more information about configuring Automatic Storage Management and raw logical volumes on HP-UX 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 by using the SAM utility:

1. Run the following command as the `root` user:

```
# sam
```

2. Select the Kernel Configuration area.
3. Select the Drivers area.
4. Select the asynchronous disk driver (`asyncdsk`).
5. Select Actions, and then select Add Driver to Kernel.
6. Select List, and then select Configurable Parameters.
7. Select the `MAX_ASYNC_PORTS` parameter.
8. Select Action, and then select Modify Configurable Parameter.
9. Specify a new value for the parameter, using the following guidelines, and then click **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 message 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. Select **Actions**, and then select **Process a New Kernel**.
11. Select one of the following options, and then click **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, then 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 restart 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 required, give the device file the operating system 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`, then run 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, click **Reports** and then click **Process List**.

4. In the Process List window, select one parallel query slave process, select **Reports**, and then select **Process Open Files**.

   The list of files currently opened by the parallel query slave process is displayed.

5. In the list of open files, check for the `/dev/async` file or the 101 0x000000 mode.

   If either is in the list, then 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 run 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 selected 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.

   You can do this, for example, by entering the following command, where *`pid`* is the process ID of a parallel query slave supposed to process asynchronous I/O:

   ```
   $ tusc -p pid > tusc.output
   ```

4. After running the query, press Ctrl+c to disconnect from the process, and 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".., 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".., 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), then 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 a number 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 nonblocking 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.

Releases of Oracle Database earlier than release 8.1.7 were able to run I/O operations only from shared memory by using the HP-UX asynchronous driver. Oracle Database 10*g* runs 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, such as the database writer process, to essentially process 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 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, then Oracle uses it to run subsequent new instances of the statement. If a shared SQL area does not exist, then 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 run-time 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, then part of the private SQL area is kept in the SGA.

The `CURSOR_SPACE_FOR_TIME` initialization parameter specifies if 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 the SQL area of each active cursor is present in memory and never ages out.
- It improves application performance, because Oracle Database does not have 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 Database 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`, then you may 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 3 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 required 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 may lead processes to indicate large memory allocations, followed by an `Out of memory` error message. If this happens, then 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 percent higher than that with a larger page size setting. With the highest setting of `L`, the memory utilization can be 50 percent 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 may 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 as `true`, unless the system indicates library cache misses when running the application. If that happens, then the shared pool may be small enough to hold the SQL areas for all concurrent open cursors.
- Decrease the virtual memory data page size for Oracle Database as required. 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
  ```

CPU_COUNT Initialization Parameter and HP-UX Dynamic Processor Reconfiguration

HP-UX 11i supports dynamic run-time 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 restart 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, then Oracle adjusts the `CPU_COUNT` parameter to the correct value to reallocate its internal resources. This enables 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, then Oracle does not dynamically update these initialization parameters to account for the new number of CPUs. This may 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, then the database may 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, then no action is required.

If the number of CPUs on the system increases by a large number, for instance from 2 to 32, then 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`), then run 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`), then edit the file and specify a new value for the parameter.
2. Restart the database.

Network Information Service (NIS) external naming support

NIS external naming adapter is supported on hp-ux. To configure and use NIS external naming, refer to the "Configuring External Naming Methods" section of *Oracle Database Net Services Administrator's Guide*.

C

Administering Oracle Database on Linux

This appendix contains information about administering Oracle Database on Linux. It contains the following topics:

Extended Buffer Cache Support

Note: This section applies to Linux x86 only.

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 systems.

See Also: *Oracle Database Concepts* for more information about the extended buffer cache feature

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:

1. Run the following command as the `root` user:

```
# mount -t tmpfs shmfs -o size=8g /dev/shm
```

2. To ensure that the in-memory file system is mounted when the system restarts, add an entry in the `/etc/fstab` file similar to the following:

```
shmfs /dev/shm tmpfs 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, then 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. This enables Oracle Database to specify a larger buffer cache.

Dynamic Cache Parameters

If the extended cache feature is enabled, then 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 nondefault 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: *Oracle Database SQL Reference* for more information about the default block size used by the CREATE TABLESPACE command

Note: The default VLM window size is 512 MB. This memory size is allocated to the address space of the process. 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, run the following command:

```
$ 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, then 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
```

Using hugetlbfs on SUSE Linux Enterprise Server 9

To enable Oracle Database to use large pages (sometimes called huge pages) on SUSE Linux Enterprise Server 9, set the value of the vm.nr_hugepages kernel parameter to specify the number of large pages that you want to reserve. You must specify a sufficient number of large pages to hold the entire SGA for the database instance. To determine the required parameter value, divide the SGA size for the instance by the size of a large page, then round up the result to the nearest integer.

To determine the default large page size, run the following command:

```
# grep Hugepagesize /proc/meminfo
```

For example, if /proc/meminfo lists the large page size as 2 MB, and the total SGA size for the instance is 1.6 GB, then set the value for the vm.nr_hugepages kernel parameter to 820 (1.6 GB / 2 MB = 819.2).

Using hugetlbfs on Red Hat Enterprise Linux AS 3

To use large pages on Red Hat Enterprise Linux AS 3:

1. Determine the memory required for the large page pool.

   To determine this value, convert the size of the SGA of the instance to megabytes, and round up by 4 MB. For example, if the SGA is 2.7 GB, then the appropriate value is 2768 MB.

2. Depending on the type of your boot loader, perform one of the following procedures:

   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 as follows:

      ```
      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. Run the following command to set the kernel parameter values:

```
# sysctl -p /etc/sysctl.conf
```

5. To verify that this amount of memory was moved successfully into the large page pool, run the following command:

```
# cat /proc/meminfo
```

   The lines at the end of the display show how many memory pages were moved into the large page pool.

6. Start Oracle Database.

Increasing SGA Address Space

> **Note:** This section applies to Linux x86 only.

Depending on your distribution of Linux, apply the instructions in one of the following sections to increase the SGA address space:

- SUSE Linux Enterprise Server 9
- Red Hat Enterprise Linux AS 3

SUSE Linux Enterprise Server 9

To increase the SGA address space on SUSE Linux Enterprise Server 9:

1. Log in as the `oracle` user.
2. In the `$ORACLE_HOME/rdbms/lib` directory, run 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 run-time memory errors, then increase the hexadecimal number specified in the first command. For example, if the 0x15000000 value prevents Oracle Database from starting, then specify the value 0x20000000. Lowering this value increases the SGA address space, but could decrease the PGA address space.

3. Run the following command to determine the process ID of the `oracle` user's shell process:

```
$ echo $$
```

   The number returned is the process ID.

4. Run the following command to switch user to `root`:

```
$ su - root
```

5. Run 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. Run the `exit` command to return to the `oracle` user's shell process, and start Oracle Listener and Oracle Database.

> **Note:** All Oracle processes must get this modified mapped base value. Starting the listener from the shell that has the modified mapped base enables client connections to connect properly.

Red Hat Enterprise Linux AS 3

To increase the SGA address space on Red Hat Enterprise Linux AS 3 or 4:

1. Log in as the `oracle` user.
2. In the `$ORACLE_HOME/rdbms/lib` directory, run 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 Management uses asynchronous I/O by default. Asynchronous I/O is not supported for database files stored on NFS file systems.

Oracle Database supports kernel asynchronous I/O. This feature is disabled by default.

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 one of the following values:

Linux Distribution	Recommended Value
SUSE Linux Enterprise Server 9	SETALL
Other distributions	ASYNCH

Direct I/O Support

Direct I/O support is available and supported on Red Hat Enterprise Linux 3 and SUSE Linux Enterprise Server 9.

Note: To use direct I/O on Red Hat Enterprise Linux 3, the driver that you use must support vary I/O.

To enable direct I/O support:

- Set the `FILESYSTEMIO_OPTIONS` initialization parameter to `DIRECTIO`.
- If you are using the asynchronous I/O option, then set the `FILESYSTEMIO_OPTIONS` initialization parameter to `SETALL`.

semtimedop Support

On Red Hat Enterprise Linux 3 and SUSE Linux Enterprise Server 9, Oracle Database supports the `semtimedop()` system call (semaphores with a time limitation). To enable support for the feature, run the following command as the `oracle` user in the `$ORACLE_HOME/rdbms/lib` directory:

```
$ make -f ins_rdbms.mk smt_on
```

To disable `semtimedop()` support, run the following command as the `oracle` user in the `$ORACLE_HOME/rdbms/lib` directory:

```
$ make -f ins_rdbms.mk smt_off
```

High-Speed Network Support

Note: This section applies to Linux x86 only.

Oracle Net supports Sockets Direct protocol (SDP) over the InfiniBand network architecture on Red Hat Enterprise Linux AS 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, refer to the following document:

`http://www.oracle.com/technology/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 in this document.

D

Administering Oracle Database on Mac OS X

This appendix contains information about administering Oracle Database on Mac OS X.

Determining Available and Used Swap Space

Mac OS X dynamically creates swap files as required in the `/private/var/vm` directory. When running Oracle Database, ensure that you have at least 1 GB of available disk space on the root (`/`) file system to accommodate newly created swap files.

To determine how much disk space is available in the `/private/var/vm` directory, run the following command:

```
$ df -k /private/var/vm
```

To determine the amount of swap space currently in use, run the following command:

```
$ du -sk /private/var/vm/swapfile*
```

E

Administering Oracle Database on Solaris

This appendix contains information about administering Oracle Database on Solaris.

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 selects 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 enables 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 is put to 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. It is not required to perform any configuration tasks to use dynamic SGA.

Note:

- Do not set the `LOCK_SGA` parameter to `TRUE` in the server parameter file. If you do, then Oracle Database 10g cannot start.
- 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 is displayed in the alert log saying that the `oradism` utility is not set up correctly, then verify that the `oradism` utility is located in the `$ORACLE_HOME/bin` directory and that it has superuser privileges.

F

Administering Oracle Database on Tru64 UNIX

This appendix contains information about administering Oracle Database on Tru64 UNIX. It contains the following topics:

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 enables sophisticated applications to communicate their specific requirements 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.

The following sections provide information about directed placement optimizations:

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. Oracle Database optimizations only affect systems that are locality sensitive.
- The operating system must be Tru64 UNIX V5.1B or later.

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, then the following message is displayed:

```
Operating System Version Does not Support NUMA.
Disabling NUMA!
```

If you enable Oracle directed placement optimizations, and later, then change Tru64 UNIX to an incompatible version, then 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. Run 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), then subtract 8388608 from the total. If the `LOG_BUFFER` value is between 8388608 and 16777216 (16 MB), then 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 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, then refer to the "Restricting Oracle Database to a Subset of the Number of RADs on the System" section on page F-4. 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, which are used by Oracle, and the SEGSZ column total is

10248 MB with the LOG_BUFFER initialization parameter set to 2097152 (2 MB), then 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 may have to raise this value slightly, by 1 or 2.

If CPUs and memory are taken off-line, then Oracle Database continues to function, but loses performance.

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, then you can run a listener on each RAD. To run the listener on a particular RAD, run 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, then Oracle Database uses the RADs on the system. If the file exists, then 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, then 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, then 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 F-4.

Supporting Mixed CPU Systems

Tru64 UNIX systems using Tru64 UNIX V5.1B or later 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 may 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 replacing 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 later. 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 run a query. If a stored statistic does not match a statistic calculated by the query optimizer while it searches for the best plan, then the query optimizer may use the wrong plan to run 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. It is not required 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, refer to *Oracle Database PL/SQL Packages and Types Reference*

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 data files that are stored on AdvFS or cluster 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, then 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, then the performance of Oracle Database is reduced and spurious I/O errors may 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. It includes the following sections:

- Single Instance Requirements
- Cluster File Systems
- Tru64 UNIX V5.1B Cluster File Systems
- Disabling Direct I/O Support

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, refer to *Oracle Database Installation Guide for UNIX Systems*.

- Oracle data files stored on an AdvFS file system or in an Automatic Storage Management 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 or later in a non-clustered system environment, the AdvFS file system and direct I/O give almost all the performance of raw devices because the file system cache is not used. In addition to this, the file system enables you to more easily manage the database files.

Cluster File Systems

On V5.1B systems or later, Tru64 UNIX supports cluster 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 Cluster File 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.

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 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 is adversely affected.

If your workload has most of the attributes described in the preceding list, then disabling direct I/O support probably improves 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, then 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`, then 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. Run the following commands:

```
# mknod /dev/timedev c 15 0
# chmod a+r /dev/timedev
```

Note: The special file `/dev/timedev` remains on the system after restarting.

3. Restart the Oracle Database instance.

   The system checks for the existence of the `/dev/timedev` file only on instance startup.

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 Automatic Storage Management 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 RAC, then ensure 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 run 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 Oracle Database).
   - It does not overlap the Tru64 UNIX file system.
   - It does not use space already used by the file system.

   To determine if 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: Ensure 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, then use the `disklabel` command to partition it.

4. Run the `ls` command to view the owner and permissions of the device file. For example:

   ```
   $ ls -la
   ```

5. Ensure that the partition is owned by the Oracle software owner. If required, 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. Ensure that the partition has the correct permissions. If required, 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 run the following command:

```
$ ls -Ll datafile
```

The following output should be displayed:

```
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, run the following command from SQL*Plus:

> **Note:** The size of an Oracle data file 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, run the following command:

```
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, refer to the Tru64 UNIX documentation or run 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, then contact HP for a patch kit.

Run 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, then reproduce the problem with the original unoptimized binary. If the problem persists, then contact Oracle Support Services.

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 F–1 lists the system resources required to run Spike.

Table F–1 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, run the following command:

```
% 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, run 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 Oracle Database binary, or if the binary had been previously optimized with OM (the predecessor to Spike from HP), then the instance startup fails with an ORA-4940 error message. If the instance startup fails, then 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 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 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, ensure 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 may 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, then perform step a. If you must merge several counts files together to better represent your workload, then perform step b.

   a. Ensure 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, then 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, then merge the `oracle.Counts.`*`pid`* files from the Oracle foreground processes that use the most memory.

      To merge the `oracle.Counts.`*`pid`* files, run the following command:

      ```
      $ prof -pixie -merge oracle.Counts $ORACLE_HOME/bin/oracle \
      oracle.Addrs oracle.Counts.pid1 oracle.Counts.pid2
      ```

8. Ensure 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 may 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.

G

Using Oracle ODBC Driver

Note: Oracle ODBC Driver is supported only for the Linux x86, Linux Itanium, and Solaris SPARC 64 platforms.

This appendix provides information related to using Oracle ODBC Driver. It contains the following sections:

Features Not Supported

Oracle ODBC Driver does not support the following ODBC 3.0 features:

- Interval data types
- `SQL_C_UBIGINT` and `SQL_C_SBIGINT` C data type identifiers
- Shared connections
- Shared environments
- The `SQL_LOGIN_TIMEOUT` attribute of SQLSetConnectAttr
- The expired password option

Oracle ODBC Driver does not support the SQL functions listed in the following table.

String Functions	Numeric Functions	Time, Date, and Interval Functions
BIT_LENGTH	ACOS	CURRENT_DATE
CHAR_LENGTH	ASIN	CURRENT_TIME
CHARACTER_LENGTH	ATAN	CURRENT_TIMESTAMP
DIFFERENCE	ATAN2	EXTRACT
OCTET_LENGTH	COT	TIMESTAMPDIFF
POSITION	DEGREES	
	RADIANS	
	RAND	
	ROUND	

Implementation of Data Types

This section discusses the DATE, TIMESTAMP, and floating point data types.

DATE and TIMESTAMP

The semantics of Oracle DATE and TIMESTAMP data types do not correspond exactly with the ODBC data types with the same names. The Oracle DATE data type contains both date and time information. The SQL_DATE data type contains only date information. The Oracle TIMESTAMP data type also contains date and time information, but it has greater precision in fractional seconds. Oracle ODBC Driver reports the data types of both Oracle DATE and TIMESTAMP columns as SQL_TIMESTAMP to prevent information loss. Similarly, Oracle ODBC Driver binds SQL_TIMESTAMP parameters as Oracle TIMESTAMP values.

> **See Also:** "DATE and TIMESTAMP Data Types" on page G-23 for information about the DATE and TIMESTAMP data types related to performance and tuning

Floating Point Data Types

When connected to a release 10.1 or later Oracle Database, Oracle ODBC Driver maps the Oracle floating point data types BINARY_FLOAT and BINARY_DOUBLE to the ODBC data types SQL_REAL and SQL_DOUBLE, respectively. In earlier releases, SQL_REAL and SQL_DOUBLE mapped to the generic Oracle numeric data type.

Limitations on Data Types

Oracle ODBC Driver and Oracle Database impose limitations on data types. The following table describes these limitations.

Limited Data Type	Description
Literals	Oracle Database limits literals in SQL statements to 4000 bytes.
SQL_LONGVARCHAR and SQL_WLONGVARCHAR	The Oracle limit for SQL_LONGVARCHAR data, where the column type is LONG, is 2,147,483,647 bytes. The Oracle limit for SQL_LONGVARCHAR data, where the column type is CLOB, is 4 gigabytes. The limiting factor is the client workstation memory.

Limited Data Type	Description
SQL_LONGVARCHAR and SQL_LONGVARBINARY	Oracle Database permits only a single long data column in each table. The long data types are SQL_LONGVARCHAR (LONG) and SQL_LONGVARBINARY (LONG RAW). Oracle recommends that you use CLOB and BLOB columns instead. There is no restriction on the number of CLOB and BLOB columns in a table.

Format of the Connection String for the SQLDriverConnect Function

The SQLDriverConnect function is one of the functions implemented by Oracle ODBC Driver. The following table describes the keywords that you can include in the connection string argument of the SQLDriverConnect function call.

Keyword	Meaning	Value
DSN	ODBC data source name	User-supplied name This is a mandatory keyword.
DBQ	TNS service name	User-supplied name This is a mandatory keyword.
UID	User ID or user name	User-supplied name This is a mandatory keyword.
PWD	Password	User-supplied name Specify PWD=; for an empty password. This is a mandatory keyword.
DBA	Database attribute	W implies write access R implies read-only access
APA	Applications attributes	T implies that thread safety is to be enabled F implies that thread safety is to be disabled
RST	Result sets	T implies that result sets are to be enabled. F implies that result sets are to be disabled.
QTO	Query timeout option	T implies that query timeout is to be enabled. F implies that query timeout is to be disabled.
CSR	Close cursor	T implies that close cursor is to be enabled. F implies that close cursor is to be disabled.
BAM	Batch autocommit mode	IfAllSuccessful implies commit only if all statements are successful (old behavior). UpToFirstFailure implies commit up to first failing statement. This is ODBC version 7 behavior. AllSuccessful implies commit all successful statements.
FBS	Fetch buffer size	User-supplied numeric value (specify a value in bytes of 0 or greater). The default is 60,000 bytes.
FEN	Failover	T implies failover is to be enabled. F implies failover is to be disabled.

Keyword	Meaning	Value
FRC	Failover retry count	User-supplied numeric value. The default is 10.
FDL	Failover delay	User-supplied numeric value. The default is 10.
LOB	LOB writes	T implies LOBs are to be enabled. F implies LOBs are to be disabled.
FWC	Force SQL_WCHAR support	T implies Force SQL_WCHAR is to be enabled. F implies Force SQL_WCHAR is to be disabled.
EXC	EXEC syntax	T implies EXEC Syntax is to be enabled. F implies EXEC Syntax is to be disabled.
XSM	Schema field	Default implies that the default value is to be used. Database implies that the Database Name is to be used. Owner implies that the name of the owner is to be used.
MDI	Set metadata ID default	T implies that the default value of SQL_ATTR_METADATA_ID is SQL_TRUE. F implies that the default value of SQL_ATTR_METADATA_ID is SQL_FALSE.
DPM	Disable SQLDescribeParam	T implies that SQLDescribeParam is to be disabled. F implies that SQLDescribeParam is to be enabled.
BTD	Bind TIMESTAMP as DATE	T implies that SQL_TIMESTAMP is to be bound as Oracle DATE. F implies that SQL_TIMESTAMP is to be bound as Oracle TIMESTAMP.
NUM	Numeric settings	NLS implies that the Globalization Support numeric settings are to be used (to determine the decimal and group separator).

Reducing Lock Timeout in a Program

Oracle Database waits indefinitely for lock conflicts between transactions to be resolved. However, you can limit the amount of time that Oracle Database waits for locks to be resolved. To do this, set the SQL_ATTR_QUERY_TIMEOUT attribute of the ODBC SQLSetStmtAttr function while calling this function before connecting to the data source.

Linking ODBC Applications

When you link your program, you must link it with the Driver Manager library, libodbc.so.

Obtaining Information About ROWIDs

The ODBC SQLSpecialColumns function returns information about the columns in a table. When used with Oracle ODBC Driver, it returns information about the Oracle ROWIDs associated with an Oracle table.

ROWIDs in a WHERE Clause

ROWIDs may be used in the `WHERE` clause of an SQL statement. However, the ROWID value must be presented in a parameter marker.

Enabling Result Sets

Oracle reference cursors, also known as result sets, enable an application to retrieve data using stored procedures and stored functions. The following information describes how to use reference cursors to enable result sets through ODBC:

- You must use the ODBC syntax for calling stored procedures. Native PL/SQL is not supported through ODBC. The following code sample identifies how to call the procedure or function without a package and within a package. The package name in this case is `RSET`.

  ```
  Procedure call:
  {CALL Example1(?)}
  {CALL RSET.Example1(?)}
  Function Call:
  {? = CALL Example1(?)}
  {? = CALL RSET.Example1(?)}
  ```

- The PL/SQL reference cursor parameters are omitted when calling the procedure. For example, assume procedure `Example2` is defined to have four parameters. Parameters 1 and 3 are reference cursor parameters and parameters 2 and 4 are character strings. The call is specified as:

  ```
  {CALL RSET.Example2("Literal 1", "Literal 2")}
  ```

The following sample application shows how to return a result set by using Oracle ODBC Driver:

```
/*
* Sample Application using Oracle reference cursors through ODBC
*
* Assumptions:
*
* 1) Oracle Sample database is present with data loaded for the EMP table.
*
* 2) Two fields are referenced from the EMP table, ename and mgr.
*
* 3) A data source has been setup to access the sample database.
*
*
* Program Description:
*
* Abstract:
*
* This program demonstrates how to return result sets using
* Oracle stored procedures
*
* Details:
*
* This program:
* Creates an ODBC connection to the database.
* Creates a Packaged Procedure containing two result sets.
* Executes the procedure and retrieves the data from both result sets.
* Displays the data to the user.
* Deletes the package then logs the user out of the database.
```

```
*
*
* The following is the actual PL/SQL this code generates to
* create the stored procedures.
*
DROP PACKAGE   ODBCRefCur;
CREATE PACKAGE ODBCRefCur AS
     TYPE ename_cur IS REF CURSOR;
     TYPE mgr_cur   IS REF CURSOR;
PROCEDURE EmpCurs(Ename IN OUT ename_cur, Mgr IN OUT mgr_cur, pjob IN VARCHAR2);

END;

/
CREATE PACKAGE BODY ODBCRefCur AS
PROCEDURE EmpCurs(Ename IN OUT ename_cur, Mgr IN OUT mgr_cur, pjob IN VARCHAR2)
     AS
       BEGIN
          IF NOT Ename%ISOPEN
          THEN
              OPEN Ename for SELECT ename from emp;
          END IF;

          IF NOT Mgr%ISOPEN
          THEN
              OPEN Mgr for SELECT mgr from emp where job = pjob;
          END IF;
       END;
END;
/


*
*  End PL/SQL for Reference Cursor.
*/

/*
 * Include Files
 */
#include <stdio.h>
#include <sql.h>
#include <sqlext.h>
/*
 * Defines
 */
#define JOB_LEN 9
#define DATA_LEN 100
#define SQL_STMT_LEN 500
/*
 * Procedures
 */
void DisplayError( SWORD HandleType, SQLHANDLE hHandle, char *Module );
/*
 * Main Program
 */
int main()
{
SQLHENV hEnv;
SQLHDBC hDbc;
SQLHSTMT hStmt;
```

```
SQLRETURN rc;
char *DefUserName ="scott";
char *DefPassWord ="tiger";
SQLCHAR ServerName[DATA_LEN];
SQLCHAR *pServerName=ServerName;
SQLCHAR UserName[DATA_LEN];
SQLCHAR *pUserName=UserName;
SQLCHAR PassWord[DATA_LEN];
SQLCHAR *pPassWord=PassWord;
char Data[DATA_LEN];
SQLINTEGER DataLen;
char error[DATA_LEN];
char *charptr;
SQLCHAR SqlStmt[SQL_STMT_LEN];
SQLCHAR *pSqlStmt=SqlStmt;
char *pSalesMan = "SALESMAN";
SQLINTEGER sqlnts=SQL_NTS;
/*
 * Allocate the Environment Handle
 */
rc = SQLAllocHandle( SQL_HANDLE_ENV, SQL_NULL_HANDLE, &hEnv );
if (rc != SQL_SUCCESS)
{
  printf( "Cannot Allocate Environment Handle\n");
  printf( "\nHit Return to Exit\n");
  charptr = gets ((char *)error);
  exit(1);
}
/*
 * Set the ODBC Version
 */
rc = SQLSetEnvAttr( hEnv,SQL_ATTR_ODBC_VERSION,(void *)SQL_OV_ODBC3,0);
if (rc != SQL_SUCCESS)
{
  printf( "Cannot Set ODBC Version\n");
  printf( "\nHit Return to Exit\n");
  charptr = gets ((char *)error);
  exit(1);
}
/*
 * Allocate the Connection handle
 */
rc = SQLAllocHandle( SQL_HANDLE_DBC, hEnv, &hDbc );
if (rc != SQL_SUCCESS)
{
  printf( "Cannot Allocate Connection Handle\n");
  printf( "\nHit Return to Exit\n");
  charptr = gets ((char *)error);
  exit(1);
}
/*
 * Get User Information
 */
strcpy ((char *) pUserName, DefUserName );
strcpy ((char *) pPassWord, DefPassWord );
/*
 * Data Source name
 */
printf( "\nEnter the ODBC Data Source Name\n" );
charptr = gets ((char *) ServerName);
```

```
/*
 * User Name
 */
printf ( "\nEnter User Name Default [%s]\n", pUserName);
charptr = gets ((char *) UserName);
if (*charptr == '\0')
{
  strcpy ((char *) pUserName, (char *) DefUserName );
}
/*
 * Password
 */
printf ( "\nEnter Password Default [%s]\n", pPassWord);
charptr = gets ((char *)PassWord);
if (*charptr == '\0')
{
  strcpy ((char *) pPassWord, (char *) DefPassWord );
}
/*
 * Connection to the database
 */
rc = SQLConnect( hDbc,pServerName,(SQLSMALLINT) strlen((char
*)pServerName),pUserName,(SQLSMALLINT) strlen((char
*)pUserName),pPassWord,(SQLSMALLINT) strlen((char *)pPassWord));
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_DBC, hDbc, "SQLConnect");
}
/*
 * Allocate a Statement
 */
rc = SQLAllocHandle( SQL_HANDLE_STMT, hDbc, &hStmt );
if (rc != SQL_SUCCESS)
{
  printf( "Cannot Allocate Statement Handle\n");
  printf( "\nHit Return to Exit\n");
  charptr = gets ((char *)error);
  exit(1);
}
/*
 * Drop the Package
 */
strcpy( (char *) pSqlStmt, "DROP PACKAGE ODBCRefCur");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
/*
 * Create the Package Header
 */
strcpy( (char *) pSqlStmt, "CREATE PACKAGE ODBCRefCur AS\n");
strcat( (char *) pSqlStmt, " TYPE ename_cur IS REF CURSOR;\n");
strcat( (char *) pSqlStmt, " TYPE mgr_cur IS REF CURSOR;\n\n");
strcat( (char *) pSqlStmt, " PROCEDURE EmpCurs (Ename IN OUT ename_cur,");
strcat( (char *) pSqlStmt, "Mgr IN OUT mgr_cur,pjob IN VARCHAR2);\n\n");
strcat( (char *) pSqlStmt, "END;\n");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Create the Package Body
```

```
 */
strcpy( (char *) pSqlStmt, "CREATE PACKAGE BODY ODBCRefCur AS\n");
strcat( (char *) pSqlStmt, " PROCEDURE EmpCurs (Ename IN OUT ename_cur,");
strcat( (char *) pSqlStmt, "Mgr IN OUT mgr_cur, pjob IN VARCHAR2)\n AS\n
BEGIN\n");
strcat( (char *) pSqlStmt, " IF NOT Ename%ISOPEN\n THEN\n");
strcat( (char *) pSqlStmt, " OPEN Ename for SELECT ename from emp;\n");
strcat( (char *) pSqlStmt, " END IF;\n\n");
strcat( (char *) pSqlStmt, " IF NOT Mgr%ISOPEN\n THEN\n");
strcat( (char *) pSqlStmt, " OPEN Mgr for SELECT mgr from emp where job =
pjob;\n");
strcat( (char *) pSqlStmt, " END IF;\n");
strcat( (char *) pSqlStmt, " END;\n");
strcat( (char *) pSqlStmt, "END;\n");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Bind the Parameter
 */
rc =
SQLBindParameter(hStmt,1,SQL_PARAM_INPUT,SQL_C_CHAR,SQL_CHAR,JOB_LEN,0,pSalesMan,0
,&sqlnts);
/*
 * Call the Store Procedure which executes the Result Sets
 */
strcpy( (char *) pSqlStmt, "{CALL ODBCRefCur.EmpCurs(?)}");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Bind the Data
 */
rc = SQLBindCol( hStmt,1,SQL_C_CHAR,Data,sizeof(Data),&DataLen);
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLBindCol");
}
/*
 * Get the data for Result Set 1
 */
printf( "\nEmployee Names\n\n");
while ( rc == SQL_SUCCESS )
{
  rc = SQLFetch( hStmt );
  if ( rc == SQL_SUCCESS )
  {
    printf("%s\n", Data);
  }
  else
  {
    if (rc != SQL_NO_DATA)
    {
      DisplayError(SQL_HANDLE_STMT, hStmt, "SQLFetch");
    }
  }
```

```
}
printf( "\nFirst Result Set - Hit Return to Continue\n");
charptr = gets ((char *)error);
/*
 * Get the Next Result Set
 */
rc = SQLMoreResults( hStmt );
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLMoreResults");
}
/*
 * Get the data for Result Set 2
 */
printf( "\nManagers\n\n");
while ( rc == SQL_SUCCESS )
{
  rc = SQLFetch( hStmt );
  if ( rc == SQL_SUCCESS )
  {
    printf("%s\n", Data);
  }
  else
  {
    if (rc != SQL_NO_DATA)
    {
    DisplayError(SQL_HANDLE_STMT, hStmt, "SQLFetch");
    }
  }
}
printf( "\nSecond Result Set - Hit Return to Continue\n");
charptr = gets ((char *)error);
/*
 * Should Be No More Results Sets
 */
rc = SQLMoreResults( hStmt );
if (rc != SQL_NO_DATA)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLMoreResults");
}
/*
 * Drop the Package
 */
strcpy( (char *) pSqlStmt, "DROP PACKAGE ODBCRefCur");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
/*
 * Free handles close connections to the database
 */
SQLFreeHandle( SQL_HANDLE_STMT, hStmt );
SQLDisconnect( hDbc );
SQLFreeHandle( SQL_HANDLE_DBC, hDbc );
SQLFreeHandle( SQL_HANDLE_ENV, hEnv );
printf( "\nAll Done - Hit Return to Exit\n");
charptr = gets ((char *)error);
return(0);
}
/*
 * Display Error Messages
 */
void DisplayError( SWORD HandleType, SQLHANDLE hHandle, char *Module )
```

```
{
SQLCHAR MessageText[255];
SQLCHAR SQLState[80];
SQLRETURN rc=SQL_SUCCESS;
long NativeError;
SWORD RetLen;
SQLCHAR error[25];
char *charptr;
rc =
SQLGetDiagRec(HandleType,hHandle,1,SQLState,&NativeError,MessageText,255,&RetLen);
printf( "Failure Calling %s\n", Module );
if (rc == SQL_SUCCESS || rc == SQL_SUCCESS_WITH_INFO)
{
  printf( "\t\t\t State: %s\n", SQLState);
  printf( "\t\t\t Native Error: %d\n", NativeError );
  printf( "\t\t\t Error Message: %s\n", MessageText );
}
printf( "\nHit Return to Exit\n");
charptr = gets ((char *)error);
exit(1);
}
```

Enabling EXEC Syntax

If the syntax of your SQL Server `EXEC` statement can be readily translated to an equivalent Oracle procedure call without requiring any change to it, then Oracle ODBC Driver can translate it if you enable this option.

The complete name of a SQL Server procedure consists of up to four identifiers:

- Server name
- Database name
- Owner name
- Procedure name

The format for the name is:

```
[[[server.][database].][owner_name].]procedure_name
```

During the migration of Microsoft SQL Server database to Oracle Database, the definition of each SQL Server procedure or function is converted to its equivalent Oracle Database syntax and is defined in a schema in Oracle Database. Migrated procedures are often reorganized (and created in schemas) in one of the following ways:

- All procedures are migrated to one schema (the default option).
- All procedures defined in one SQL Server database are migrated to the schema named with that database name.
- All procedures owned by one user are migrated to the schema named with that user's name.

To support these three ways of organizing migrated procedures, you can specify one of these schema name options for translating procedure names. Object names in the translated Oracle procedure call are not case-sensitive.

Supported Functionality

This sections provides information about the functionality supported by Oracle ODBC Driver. It contains the following sections:

API Conformance

Oracle ODBC Driver release 10.2.0.1.0 and higher supports all Core, Level 2, and Level1 functions.

Implementation of ODBC API Functions

The following table describes how Oracle ODBC Driver implements specific functions.

Function	Description
`SQLConnect`	`SQLConnect` requires only a `DBQ`, user ID, and password.
`SQLDriverConnect`	`SQLDriverConnect` uses the `DSN`, `DBQ`, `UID`, and `PWD` keywords.
`SQLSpecialColumns`	If `SQLSpecialColumns` is called with the `SQL_BEST_ROWID` attribute, it always returns the ROWID column.
`SQLProcedures` and `SQLProcedureColumns`	Refer to the information in the following row.
All catalog functions	If the `SQL_ATTR_METADATA_ID` statement attribute is set to `SQL_TRUE`, then a string argument is treated as an identifier argument, and its case is not significant. In this case, the underscore (_) and the percent sign (%) are treated as the actual character, and not as a search pattern character. In contrast, if this attribute is set to `SQL_FALSE`, then it is either an ordinary argument or a pattern value argument and is treated literally, and its case is significant.

SQLProcedures and SQLProcedureColumns

The `SQLProcedures` and `SQLProcedureColumns` calls have been modified to locate and return information about all procedures and functions even if they are contained within a package. In earlier releases, the calls only found procedures and functions that were outside of packages. The following examples and scenarios show what procedures or functions are returned if the `SQL_ATTR_METADATA_ID` attribute is set to `SQL_FALSE`.

Suppose that you have the following stored procedures:

```
"BAR"
"BARX"
"XBAR"
"XBARX"
"SQLPROCTEST.BAR"
"SQLPROCTEST.BARX"
"SQLPROCTEST.XBAR"
"SQLPROCTEST.XBARX"
```

When you look for % or %%%%%%, you get all eight procedures.

When you look for %_ or _%, you get the following:

```
BAR
BARX
XBAR
XBARX
```

When you look for . or .% or %.% or SQLPROC%. or SQLPROC%.%, you get the following:

```
SQLPROCTEST.BAR
SQLPROCTEST.BARX
SQLPROCTEST.XBAR
SQLPROCTEST.XBARX
```

When you look for %BAR, you get the following:

```
BAR
XBAR
```

When you look for .%BAR or %.%BAR, you get the following:

```
SQLPROCTEST.BAR
SQLPROCTEST.XBAR
```

When you look for SQLPROC% or .SQLPROC%, you get the following:

```
nothing (0 rows)
```

Implementation of the ODBC SQL Syntax

If a comparison predicate has a parameter marker as the second expression in the comparison and the value of that parameter is set to SQL_NULL_DATA with SQLBindParameter, then the comparison fails. This is consistent with the null predicate syntax in ODBC SQL.

Implementation of Data Types

For programmers, the most important part of the implementation of the data types concerns the CHAR, VARCHAR, and VARCHAR2 data types.

For an fSqlType value of SQL_VARCHAR, SQLGetTypeInfo returns the Oracle Database data type VARCHAR2. For an fSqlType value of SQL_CHAR, SQLGetTypeInfo returns the Oracle Database data type CHAR.

Unicode Support

This section provide information about Unicode support. It contains the following topics:

- Unicode Support Within the ODBC Environment
- Unicode Support in ODBC API
- SQLGetData Performance
- Unicode Samples

Unicode Support Within the ODBC Environment

ODBC Driver Manager makes all ODBC drivers, regardless of whether or not they support Unicode, appear as if they are Unicode compliant. This allows ODBC applications to be written independent of the Unicode capabilities of underlying ODBC drivers.

The extent to which the Driver Manager can emulate Unicode support for ANSI ODBC drivers is limited by the conversions possible between the Unicode data and the local code page. Data loss is possible when the Driver Manager is converting from Unicode to the local code page. Full Unicode support is not possible unless the underlying ODBC driver supports Unicode. Oracle ODBC Driver provides full Unicode support.

Unicode Support in ODBC API

The ODBC API supports both Unicode and ANSI entry points using the `W` and `A` suffix convention. An ODBC application developer does not need to explicitly call entry points with the suffix. An ODBC application that is compiled with the UNICODE and _UNICODE preprocessor definitions will generate the appropriate calls. For example, a call to `SQLPrepare` will be compiled as `SQLPrepareW`.

The C data type, `SQL_C_WCHAR`, was added to the ODBC interface to allow applications to specify that an input parameter is encoded as Unicode or to request column data returned as Unicode. The macro `SQL_C_TCHAR` is useful for applications that need to be built as both Unicode and ANSI. The `SQL_C_TCHAR` macro compiles as `SQL_C_WCHAR` for Unicode applications and as `SQL_C_CHAR` for ANSI applications.

The SQL data types, `SQL_WCHAR`, `SQL_WVARCHAR`, and `SQL_WLONGVARCHAR`, have been added to the ODBC interface to represent columns defined in a table as Unicode. Potentially, these values are returned from calls to `SQLDescribeCol`, `SQLColAttribute`, `SQLColumns`, and `SQLProcedureColumns`.

Unicode encoding is supported for SQL column types NCHAR, NVARCHAR2, and NCLOB. In addition, Unicode encoding is also supported for SQL column types CHAR and VARCHAR2 if the character semantics are specified in the column definition.

Oracle ODBC Driver supports these SQL column types and maps them to ODBC SQL data types. The following table lists the supported SQL data types and the equivalent ODBC SQL data type.

SQL Data Types	ODBC SQL Data Types
CHAR	SQL_CHAR or SQL_WCHAR
VARCHAR2	SQL_VARCHAR or SQL_WVARCHAR
NCHAR	SQL_WCHAR
NVARCHAR2	SQL_WVARCHAR
NCLOB	SQL_WLONGVARCHAR

SQLGetData Performance

The `SQLGetData` function allows an ODBC application to specify the data type to receive a column as after the data has been fetched. OCI requires Oracle ODBC Driver to specify the data type before it is fetched. In this case, Oracle ODBC Driver uses information about the data type of the column (as defined in the database) to determine how to best default to fetching the column through OCI.

If a column that contains character data is not bound by SQLBindCol, then Oracle ODBC Driver needs to determine if it should fetch the column as Unicode or as the local code page. The driver could always default to receiving the column as Unicode. However, this may result in as many as two unnecessary conversions. For example, if the data were encoded in the database as ANSI, there would be an ANSI to Unicode conversion to fetch the data into Oracle ODBC Driver. If the ODBC application then requested the data as SQL_C_CHAR, there would be an additional conversion to revert the data back to its original encoding.

The default encoding of Oracle Database Client is used when fetching data. However, an ODBC application may overwrite this default and fetch the data as Unicode by binding the column or the parameter as the WCHAR data type.

Unicode Samples

Because Oracle ODBC Driver itself was implemented using TCHAR macros, it is recommended that ODBC application programs use TCHAR in order to take advantage of the driver.

The following examples show how to use TCHAR, which becomes the WCHAR data type if you compile with UNICODE and _UNICODE.

Example G–1 Connection to Database

To use this code, you only need to specify the Unicode literals for SQLConnect.

```
HENV         envHnd;
HDBC         conHnd;
HSTMT        stmtHnd;
RETCODE      rc;

rc  = SQL_SUCCESS;

// ENV is allocated
rc = SQLAllocEnv(&envHnd);
// Connection Handle is allocated
rc = SQLAllocConnect(envHnd, &conHnd);
rc = SQLConnect(conHnd, _T("stpc19"), SQL_NTS, _T("scott"), SQL_NTS, _T("tiger"),
SQL_NTS);
.
.
.
if (conHnd)
  SQLFreeConnect(conHnd);
if (envHnd)
  SQLFreeEnv(envHnd);
```

Example G–2 Simple Retrieval

The following example retrieves the employee names and the job tiles from the EMP table. With the exception that you need to specify TCHAR compliant data to every ODBC function, there is no difference to the ANSI case. If the case is a Unicode application, then you have to specify the length of the buffer to the BYTE length when you call SQLBindCol. For example, sizeof(ename).

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLExecDirect
```

```
**   SQLBindCol
**   SQLFetch
**
*/
static SQLTCHAR *sqlStmt = _T("SELECT ename, job FROM emp");
SQLTCHAR   ename[50];
SQLTCHAR     job[50];
SQLINTEGER enamelen, joblen;

_tprintf(_T("Retrieve ENAME and JOB using SQLBindCol 1.../n[%s]/n"), sqlStmt);

// Step 1: Prepare and Execute
rc = SQLExecDirect(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Bind Columns
rc = SQLBindCol(stmtHnd,
                1,
                SQL_C_TCHAR,
                ename,
                sizeof(ename),
                &enamelen);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindCol(stmtHnd,
                2,
                SQL_C_TCHAR,
                job,
                sizeof(job),
                &joblen);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

do
{
  // Step 3: Fetch Data
  rc = SQLFetch(stmtHnd);
  if (rc == SQL_NO_DATA)
    break;
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);
  _tprintf(_T("ENAME = %s, JOB = %s/n"), ename, job);
} while (1);
_tprintf(_T("Finished Retrieval/n/n"));
```

Example G–3 Retrieval Using SQLGetData (Binding After Fetch)

This example shows how to use `SQLGetData`. There is no difference to the ANSI application in terms of Unicode-specific issues.

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLExecDirect
**   SQLFetch
**   SQLGetData
*/
static SQLTCHAR *sqlStmt = _T("SELECT ename,job FROM emp"); // same as Case 1.
SQLTCHAR         ename[50];
SQLTCHAR         job[50];

_tprintf(_T("Retrieve ENAME and JOB using SQLGetData.../n[%s]/n"), sqlStmt);
```

```
if (rc != SQL_SUCCESS)
{
  _tprintf(_T("Failed to allocate STMT/n"));
  goto exit2;
}

// Step 1: Prepare and Execute
rc = SQLExecDirect(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

do
{

  // Step 2: Fetch
  rc = SQLFetch(stmtHnd);
  if (rc == SQL_NO_DATA)
    break;
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);

  // Step 3: GetData
  rc = SQLGetData(stmtHnd,
    1,
    SQL_C_TCHAR,
    (SQLPOINTER)ename,
    sizeof(ename),
    NULL);
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);
  rc = SQLGetData(stmtHnd,
    2,
    SQL_C_TCHAR,
    (SQLPOINTER)job,
    sizeof(job),
    NULL);
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);
  _tprintf(_T("ENAME = %s, JOB = %s/n"), ename, job);
} while (1);
_tprintf(_T("Finished Retrieval/n/n"));
```

Example G–4 Simple Update

This example shows how to update data. The length of data for `SQLBindParameter` has to be specified with the BYTE length, even in the case of a Unicode application.

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLPrepare
**   SQLBindParameter
**   SQLExecute
*/
static SQLTCHAR *sqlStmt = _T("INSERT INTO emp(empno,ename,job) VALUES(?,?,?)");
static SQLTCHAR *empno   = _T("9876");      // Emp No
static SQLTCHAR *ename   = _T("ORACLE");    // Name
static SQLTCHAR *job     = _T("PRESIDENT"); // Job

_tprintf(_T("Insert User ORACLE using SQLBindParameter.../n[%s]/n"), sqlStmt);

// Step 1: Prepare
rc = SQLPrepare(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
```

```
// Step 2: Bind Parameter
rc = SQLBindParameter(stmtHnd,
                      1,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_DECIMAL,
                      4,               // 4 digit
                      0,
                      (SQLPOINTER)empno,
                      0,
                      NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindParameter(stmtHnd,
                      2,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_CHAR,
                      lstrlen(ename)*sizeof(TCHAR),
                      0,
                      (SQLPOINTER)ename,
                      lstrlen(ename)*sizeof(TCHAR),
                      NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindParameter(stmtHnd,
                      3,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_CHAR,
                      lstrlen(job)*sizeof(TCHAR),
                      0,
                      (SQLPOINTER)job,
                      lstrlen(job)*sizeof(TCHAR),
                      NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 3: Execute
rc = SQLExecute(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
```

Example G–5 Update and Retrieval for Long Data (CLOB)

This example may be the most complicated case to update and retrieve data for long data, like `CLOB`, in Oracle Database. Because the length of data should always be the `BYTE` length, the expression `lstrlen(TCHAR data)*sizeof(TCHAR)` is needed to derive the `BYTE` length.

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLPrepare
**   SQLBindParameter
**   SQLExecute
**   SQLParamData
**   SQLPutData
**
**   SQLExecDirect
**   SQLFetch
```

```
**   SQLGetData
*/
static SQLTCHAR *sqlStmt1  = _T("INSERT INTO clobtbl(clob1) VALUES(?)");
static SQLTCHAR *sqlStmt2  = _T("SELECT clob1 FROM clobtbl");
SQLTCHAR         clobdata[1001];
SQLTCHAR         resultdata[1001];
SQLINTEGER       ind = SQL_DATA_AT_EXEC;
SQLTCHAR        *bufp;
int              clobdatalen, chunksize, dtsize, retchklen;

_tprintf(_T("Insert CLOB1 using SQLPutData.../n[%s]/n"), sqlStmt1);

// Set CLOB Data
{
  int i;
  SQLTCHAR ch;
  for (i=0, ch=_T('A'); i< sizeof(clobdata)/sizeof(SQLTCHAR); ++i, ++ch)
  {
    if (ch > _T('Z'))
      ch = _T('A');
    clobdata[i] = ch;
  }
  clobdata[sizeof(clobdata)/sizeof(SQLTCHAR)-1] = _T('/0');
}
clobdatalen = lstrlen(clobdata);  // length of characters
chunksize   = clobdatalen / 7;    // 7 times to put

// Step 1: Prepare
rc = SQLPrepare(stmtHnd, sqlStmt1, SQL_NTS);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Bind Parameter with SQL_DATA_AT_EXEC
rc = SQLBindParameter(stmtHnd,
                      1,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_LONGVARCHAR,
                      clobdatalen*sizeof(TCHAR),
                      0,
                      (SQLPOINTER)clobdata,
                      clobdatalen*sizeof(TCHAR),
                      &ind);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
// Step 3: Execute
rc = SQLExecute(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 4: ParamData (initiation)
rc = SQLParamData(stmtHnd, (SQLPOINTER*)&bufp); // set value
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

for (dtsize=0, bufp = clobdata;
     dtsize < clobdatalen;
     dtsize += chunksize, bufp += chunksize)
{
  int len;
  if (dtsize+chunksize<clobdatalen)
    len = chunksize;
  else
    len = clobdatalen-dtsize;
```

```
  // Step 5: PutData
  rc = SQLPutData(stmtHnd, (SQLPOINTER)bufp, len*sizeof(TCHAR));
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);
}

// Step 6: ParamData (temination)
rc = SQLParamData(stmtHnd, (SQLPOINTER*)&bufp);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLFreeStmt(stmtHnd, SQL_CLOSE);
_tprintf(_T("Finished Update/n/n"));
rc = SQLAllocStmt(conHnd, &stmtHnd);
if (rc != SQL_SUCCESS)
{
  _tprintf(_T("Failed to allocate STMT/n"));
  goto exit2;
}

// Clear Result Data
memset(resultdata, 0, sizeof(resultdata));
chunksize   = clobdatalen / 15;   // 15 times to put

// Step 1: Prepare
rc = SQLExecDirect(stmtHnd, sqlStmt2, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Fetch
rc = SQLFetch(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

for(dtsize=0, bufp = resultdata;
    dtsize < sizeof(resultdata)/sizeof(TCHAR) && rc != SQL_NO_DATA;
    dtsize += chunksize-1, bufp += chunksize-1)
{
  int len; // len should contain the space for NULL termination
  if (dtsize+chunksize<sizeof(resultdata)/sizeof(TCHAR))
    len = chunksize;
  else
    len = sizeof(resultdata)/sizeof(TCHAR)-dtsize;

  // Step 3: GetData
  rc = SQLGetData(stmtHnd,
    1,
    SQL_C_TCHAR,
    (SQLPOINTER)bufp,
    len*sizeof(TCHAR),
    &retchklen);
}
if (!_tcscmp(resultdata, clobdata))
{
  _tprintf(_T("Succeeded!!/n/n"));
}
else
{
  _tprintf(_T("Failed!!/n/n"));
}
```

Performance and Tuning

This section contains the following topics:

General ODBC Programming Guidelines

Apply the following programming guidelines to improve the performance of an ODBC application:

- Enable connection pooling if the application will frequently connect and disconnect from a data source. Reusing pooled connections is extremely efficient compared to reestablishing a connection.
- Minimize the number of times a statement needs to be prepared. Where possible, use bind parameters to make a statement reusable for different parameter values. Preparing a statement once and running it several times is much more efficient than preparing the statement for every `SQLExecute`.
- Do not include columns in a `SELECT` statement of which you know the application will not retrieve; especially `LONG` columns. Because of the nature of the database server protocols, Oracle ODBC Driver must fetch the entire contents of a `LONG` column if it is included in the `SELECT` statement, regardless of whether the application binds the column or performs a `SQLGetData` operation.
- If you are performing transactions that do not update the data source, then set the `SQL_ATTR_ACCESS_MODE` attribute of the ODBC `SQLSetConnectAttr` function to `SQL_MODE_READ_ONLY`.
- If you are not using ODBC escape clauses, then set the `SQL_ATTR_NOSCAN` attribute of the ODBC `SQLSetConnectAttr` function or the ODBC `SQLSetStmtAttr` function to true.
- Use the ODBC `SQLFetchScroll` function instead of the ODBC `SQLFetch` function for retrieving data from tables that have a large number of rows.

Data Source Configuration Options

This section discusses the performance implications of the following ODBC data source configuration options:

- Enable Result Sets

  This option enables the support of returning result sets (for example, `RefCursor`) from procedure calls. The default is enabling the returning of result sets.

  Oracle ODBC Driver must query the database server to determine the set of parameters for a procedure and their data types in order to determine if there are any `RefCursor` parameters. This query incurs an additional network round trip the first time any procedure is prepared and executed.

- Enable LOBs

  This option enables the support of inserting and updating LOBs. The default is enabled.

  Oracle ODBC Driver must query the database server to determine the data types of each parameter in an `INSERT` or `UPDATE` statement to determine if there are

any LOB parameters. This query incurs an additional network round trip the first time any INSERT or UPDATE is prepared and run.

- Bind TIMESTAMP as DATE

  Binds SQL_TIMESTAMP parameters as the appropriate Oracle Database data type. If this option is set to TRUE, then SQL_TIMESTAMP binds as the Oracle DATE data type. If this option is set to FALSE, then SQL_TIMESTAMP binds as the Oracle TIMESTAMP data type, which is the default.

- Enable Closing Cursors

  The SQL_CLOSE option of the ODBC function, SQLFreeStmt, is supposed to close associated cursors with a statement and discard all pending results. The application can reopen the cursor by running the statement again without doing a SQLPrepare again. A typical scenario for this would be an application that expects to be idle for a while but will reuse the same SQL statement again. While the application is idle, it may want to free up any associated server resources.

  The OCI, on which Oracle ODBC Driver is layered, does not support the functionality of closing cursors. Therefore, by default, the SQL_CLOSE option has no effect in Oracle ODBC Driver. The cursor and associated resources remain open on the database.

  Enabling this option causes the associated cursor to be closed on the database server. However, this results in the parse context of the SQL statement being lost. The ODBC application can run the statement again without calling SQLPrepare. However, internally, Oracle ODBC Driver must prepare and run the statement all over. Enabling this option has a severe performance impact on applications that prepare a statement once and run it repeatedly.

  This option should only be enabled if freeing the resources on the server is necessary.

- Fetch Buffer Size

  Set the Fetch Buffer Size (FetchBufferSize) in the odbc.ini file to a value specified in bytes. This value is the amount of memory needed that will determine how many rows of data Oracle ODBC Driver will pre-fetch at a time from an Oracle Database to the client's cache regardless of the number of rows the application program requests in a single query, thus improving performance.

  There will be an improvement in the response time of applications that typically fetch fewer than 20 rows of data at a time, particularly over slow network connections or from heavily loaded servers. Setting this too high can have an adverse effect on response time or consume large amounts of memory. The default is 64,000 bytes. You should choose an optimal value for your application.

  When the LONG and LOB data types are present, the number of rows pre-fetched by Oracle ODBC Driver is not determined by the Fetch Buffer Size. The inclusion of the LONG and LOB data types minimizes the performance improvement and could result in excessive memory use. Oracle ODBC Driver ignores the Fetch Buffer Size and only pre-fetches a set number of rows in the presence of the LONG and LOB data types.

  See Also: "Format of the Connection String for the SQLDriverConnect Function" on page G-3

DATE and TIMESTAMP Data Types

If a DATE column in the database is used in a WHERE clause and the column has an index, then there can be an impact on performance. For example:

```
SELECT * FROM EMP WHERE HIREDATE = ?
```

In this example, an index on the HIREDATE column could be used to make the query run quickly. However, because HIREDATE is a DATE value and Oracle ODBC Driver is supplying the parameter value as TIMESTAMP, the query optimizer of Oracle Database must apply a conversion function. To prevent incorrect results (as might happen if the parameter value had nonzero fractional seconds), the optimizer applies the conversion to the HIREDATE column resulting in the following statement:

```
SELECT * FROM EMP WHERE TO_TIMESTAMP(HIREDATE) = ?
```

However, this has the effect of disabling the use of the index on the HIREDATE column. Instead, the server performs a sequential scan of the table. If the table has many rows, then this can take a long time. As a workaround for this situation, Oracle ODBC Driver has the connection option to bind TIMESTAMP as DATE. When this option is enabled, Oracle ODBC Driver binds SQL_TIMESTAMP parameters as the Oracle DATE data type instead of the Oracle TIMESTAMP data type. This enables the query optimizer to use any index on the DATE columns.

> **Note:** This option is intended only for use with Microsoft Access or other similar programs that bind DATE columns as TIMESTAMP columns. It should not be used when there are actual TIMESTAMP columns present or when data loss may occur. Microsoft Access runs such queries using whatever columns are selected as the primary key.

Error Messages

When an error occurs, Oracle ODBC Driver returns the native error number, the SQLSTATE (an ODBC error code), and an error message. The driver derives this information both from errors detected by the driver and errors returned by Oracle Database.

Native Error

For errors that occur in the data source, Oracle ODBC Driver returns the native error returned to it by Oracle Database. When Oracle ODBC Driver or the Driver Manager detects an error, Oracle ODBC Driver returns a native error of zero.

SQLSTATE

For errors that occur in the data source, Oracle ODBC Driver maps the returned native error to the appropriate SQLSTATE. When Oracle ODBC Driver or the Driver Manager detects an error, it generates the appropriate SQLSTATE.

Error Message

For errors that occur in the data source, Oracle ODBC Driver returns an error message based on the message returned by Oracle Database. For errors that occur in Oracle ODBC Driver or the Driver Manager, Oracle ODBC Driver returns an error message based on the text associated with the SQLSTATE.

Error messages have the following format:

```
[vendor] [ODBC-component] [data-source] error-message
```

The prefixes in brackets ([ ]) identify the source of the error. The following table shows the values of these prefixes returned by Oracle ODBC Driver. When the error occurs in the data source, the `vendor` and `ODBC-component` prefixes identify the vendor and name of the ODBC component that received the error from the data source.

Error Source	Prefix	Value
Driver Manager	[vendor]	[unixODBC]
	[ODBC-component]	[Driver Manager]
	[data-source]	Not applicable
Oracle ODBC Driver	[vendor]	[ORACLE]
	[ODBC-component]	[Oracle ODBC Driver]
	[data-source]	Not applicable
Oracle Database	[vendor]	[ORACLE]
	[ODBC-component]	[Oracle ODBC Driver]
	[data-source]	[Oracle OCI]

For example, if the error message does not contain the `Ora` prefix shown in the following format, the error is an Oracle ODBC Driver error and should be self-explanatory.

```
[Oracle][ODBC]Error message text here
```

If the error message contains the `Ora` prefix shown in the following format, then it is not an Oracle ODBC Driver error:

```
[Oracle][ODBC][Ora]Error message text here
```

> **Note:** Although the error message contains the `Ora` prefix, the actual error may originate from one of several sources.

If the error message text starts with the following prefix, then you can obtain more information about the error in Oracle Database documentation:

```
ORA-
```

H

Database Limits

This appendix describes database limits.

Database Limits

Table H–1 lists the default and maximum values for parameters in a `CREATE DATABASE` or `CREATE CONTROLFILE` statement.

> **Note:** Interdependencies between these parameters may affect permissible values.

Table H–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 H–2 lists the Oracle Database file size limits in bytes.

Table H–2 File Size Limits

File Type	Platform	
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, and Solaris: 32-bit with 32-bit files	2,147,483,647 bytes
	AIX, HP-UX, Linux, Mac OS X, and Solaris: 64-bit files	Unlimited
Control files	HP-UX, Linux, Mac OS X, and Solaris	20000 database blocks
	AIX	10000 database blocks
	Tru64 UNIX	19200 database blocks

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

X