Oracle® Database

Administrator's Reference

11*g* Release 2 (11.2) for Linux and UNIX-Based Operating Systems

E10839-02

August 2009

ORACLE®

Oracle Database Administrator's Reference, 11*g* Release 2 (11.2) for Linux and UNIX-Based Operating Systems

E10839-02

Primary Author: Namrata Bhakthavatsalam, Reema Khosla

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

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

Contents

7 SQL*Loader and PL/SQL Demonstrations

8 Tuning Oracle Database

A Administering Oracle Database on Linux

B Using Oracle ODBC Driver

Preface

This guide provides platform-specific information about administering and configuring Oracle Database 11*g* Release 2 (11.2) on the following platforms:

- AIX Based Systems
- HP-UX
- Linux
- Solaris Operating System

This guide supplements the *Oracle Database Administrator's Guide*.

Audience

This guide is intended for anyone responsible for administering and configuring Oracle Database 11*g* Release 2 (11.2). If you are configuring Oracle RAC, then refer to *Oracle Real Application Clusters Administration and Deployment Guide*.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

Related Documentation

Refer to the appropriate section for a listing of Oracle Database 11*g* documentation specific to the platform:

Linux Documentation

- Oracle Database
  - *Oracle Database Release Notes for Linux*
  - *Oracle Database Installation Guide for Linux*
  - *Oracle Database Quick Installation Guide for Linux x86*
  - *Oracle Database Quick Installation Guide for Linux x86-64*
  - *Oracle Grid Infrastructure Installation Guide for Linux*
  - *Oracle Real Application Clusters Installation Guide for Linux and UNIX*
  - *Oracle Enterprise Manager Grid Control Installation and Basic Configuration*
- Oracle Database Client
  - *Oracle Database Client Installation Guide for Linux*
  - *Oracle Database Client Quick Installation Guide for Linux x86*
  - *Oracle Database Client Quick Installation Guide for Linux x86-64*
- Oracle Database Examples
  - *Oracle Database Examples Installation Guide*

For important information that was not available when this book was released, refer to the release notes for the 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`
italic	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 Based Systems	AIX
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.
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 **Note:** Where the information for Solaris is different on a particular architecture, this is noted in the text.

Accessing Documentation

The documentation for this release includes platform-specific documentation and generic product documentation. Both the documents include information on installing, configuring and, using Oracle products on a particular platform. The documentation is available in both Adobe portable document format (PDF) and HTML format.

- To access the platform-specific documentation on installation media:
  1. Use a Web browser to open the `welcome.htm` file in the top-level directory of the installation media.
  2. For DVDs only, select the appropriate product link.
  3. Select the **Documentation** tab.
- To access the product documentation, see Oracle Technology Network Web site:

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

  Note: Platform-specific documentation is current at the time of release. For the latest information, Oracle recommends you to go to Oracle Technology Network Web site.

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
```

You can create a syslog audit trail to track administrative activities.

> **See Also:** "Using the Syslog Audit Trail to Audit System Administrators on UNIX Systems" in *Oracle Database Security Guide* for more information on audit trails.

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. This section covers the following topics:

- Oracle Database Environment Variables
- UNIX Environment Variables
- Setting a Common Environment
- Setting the System Time Zone

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 some of 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 client character set specified by `NLS_LANG` must match the character set of the terminal or terminal emulator. If required, `NLS_LANG` can be temporarily reset to another character set before starting a non-interactive batch program to match the character set of files and scripts processed by this program. 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.we8iso8859p15`
`ORA_NLS10`	Function	Specifies the directory where the language, territory, character set, and linguistic definition files are stored.
	Syntax	*`directory_path`*
	Example	`$ORACLE_HOME/nls/data`
`ORA_TZFILE`	Function	Specifies the full path and file name of the time zone file. The Oracle Database Server always uses the large time zone file (`$ORACLE_HOME/oracore/zoneinfo/timezlrg_`*`number`*`.dat`). If you want to use the small time zone file on the client side, you must set this environment variable to the full path of the small time zone file (`$ORACLE_HOME/oracore/zoneinfo/timezone_`*`number`*`.dat`). If you use the small time zone file on the client side, you must ensure that the database you access contains data only in the time zone regions recognized by the small time zone file.
	Syntax	*`directory_path`*
	Example	`$ORACLE_HOME/oracore/zoneinfo/timezlrg_11.dat`
`ORACLE_BASE`	Function	Specifies the base of the Oracle directory structure for Optimal Flexible Architecture compliant installations.
	Syntax	*`directory_path`*
	Example	`/u01/app/oracle`
`ORACLE_HOME`	Function	Specifies the directory containing the Oracle software.
	Syntax	*`directory_path`*

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

Variable	Detail	Definition
	Example	`$ORACLE_BASE/product/11.2.0/dbhome_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/11.2.0/dbhome_1/bin:.` **Note:** The period adds the current working directory to the search path.
`ORACLE_SID`	Function	Specifies the Oracle system identifier.
	Syntax	A string of numbers and letters that must begin with a letter. Oracle recommends a maximum of 8 characters for system identifiers. For more information about this environment variable, refer to *Oracle Database Installation Guide*.
	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 following command: `SQL> CONNECT USERNAME` `Enter password:` *`password`*
	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`

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

Variable	Detail	Definition
NLS_OS_CHARSET	Function	Specifies the Oracle character set name corresponding to the UNIX locale character set in which the file names and user names are encoded by the operating system. You must set the environment variable NLS_OS_CHARSET, if the UNIX locale character set is different from the Oracle client character set. These two character sets may differ, for example, if NLS_LANG is set to a particular character set used to encode an SQL script, which is to be executed in an SQL*Plus session. Usually, the Oracle client character set and the operating system character set are the same and NLS_OS_CHARSET does not need to be set.
	Syntax	*characterset*
	Example	WE8ISO8859P1

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
ORA_FPU_PRECISION (Linux x86 only)	Function	For all precompiled applications where calculations are done using the extended precision of the x86 Floating Point Unit, this variable must be set to EXTENDED before running the application. **Note**: Setting this variable results in non-IEEE compliant floating point results. Hence, ORA_FPU_PRECISION should not be set if you are using either BINARY_FLOAT or BINARY_DOUBLE datatypes, documented as note 246916.1 in My Oracle Support (formerly Oracle*MetaLink*) Web site: http://metalink.oracle.com/
CLASSPATH	Function	Used with Java applications. The required setting for this variable depends on the Java application. Refer to the product documentation for 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
	Syntax	Colon-separated list of directories: *directory1:directory2:directory3*

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

Variable	Detail	Definition
	Example	`/usr/lib:$ORACLE_HOME/lib`
`HOME`	Function	The home directory of the user.
	Syntax	*`directory_path`*
	Example	`/home/oracle`
`LANG` or `LC_ALL`	Function	Specifies the language and character set used by the operating system for messages and other output. Oracle tools that are programmed in Java, such as Oracle Universal Installer and Database Configuration Assistant, may also use this variable to determine the language of their user interface. Refer to the operating system documentation for more information.
`LD_OPTIONS`	Function	Specifies the default linker options. Refer to the `ld` man page for more information about this environment variable.
`LPDEST` (Solaris only)	Function	Specifies the name of the default printer.
	Syntax	*`string`*
	Example	`docprinter`
`LD_LIBRARY_PATH`	Function	Environment variable to specify the path used to search for libraries on UNIX and Linux. The environment variable may have a different name on some operating systems, such as `LIBPATH` on AIX, and `SHLIB_PATH` on HP-UX.
	Syntax	Colon-separated list of directories: *`directory1:directory2:directory3`*
	Example	`/usr/dt/lib:$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`
`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`

Setting a Common Environment

This section describes how to set a common operating system environment by using the `oraenv` or `coraenv` scripts, depending on the 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 the 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.

See Also: *Oracle Database Globalization Support Guide* and *Oracle Database Administrator's Guide*for more information about setting the database time zone

Initialization Parameters

The following sections provide information about Oracle Database initialization parameters:

- DB_BLOCK_SIZE Initialization Parameter
- ASM_DISKSTRING Initialization Parameter
- LOG_ARCHIVE_DEST_n Initialization Parameter

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 x86. It is 32 KB on other platforms.

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/rdisk/disk*
Solaris	/dev/rdsk/*
Linux	/dev/raw/*

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
HP-UX	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:

- Creating Additional Operating System Accounts

- Configuring the Accounts of Oracle Users

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.

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 Trace Files

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

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/diag/rdbms/mydb/mydb/trace/test_lgwr_1237.trc
```

Set the `MAX_DUMP_FILE` initialization parameter to at least 5000 to ensure that the trace file is large enough to store error information.

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

Note: When using Oracle Restart, you can use Service Control Utility (SRVCTL), a command-line interface, to manage Oracle processes (database instance, listener, ASM instance). With SRVCTL, you can manage the Oracle Restart configuration, see the status of processes managed by Oracle Restart, and start or stop processes such as the Oracle Database. SRVCTL has been enhanced to support single instance databases with Oracle Restart on standalone servers and on clusters with Oracle Clusterware.

See Also: *Oracle Database Administrator's Guide* and *Oracle Database Storage Administrator's Guide* for more information about SRVCTL commands

Stopping and Starting Oracle Processes

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

- Stopping and Starting Oracle Database and Automatic Storage Management Instances
- Stopping and Starting Oracle Restart
- Stopping and Starting Oracle Enterprise Manager Database Control
- Stopping and Starting Oracle Management Agent

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 should be 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 the default shell, run the `oraenv` or `coraenv` script to set the environment variables for the instance that should be 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
   SQL> CONNECT SYS as SYSDBA
   Enter password: sys_password
   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 to set the `ORACLE_SID` and `ORACLE_HOME` environment variables to identify the SID and Oracle home directory for the instance you want to start.
2. Run the following commands to start the instance:

```
$ sqlplus
SQL> CONNECT SYS as SYSDBA
Enter password: sys_password
SQL> STARTUP
```

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

Stopping and Starting Oracle Restart

To stop or start Oracle Restart, run the following command:

- Start: This option is used to start Oracle Restart

  Syntax and Options:

  ```
  crsctl start has
  ```

- Stop: This option is used to stop Oracle Restart

  Syntax and Options:

  ```
  crsctl stop has
  ```

See Also: *Oracle Database Administrator's Guide* for more information about the `srvctl` commands

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 the 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:

```
$ 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 the system to automatically start Oracle Database when the system starts, 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 the operating system documentation for more information about system startup and shutdown procedures

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 the platform.

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

   - On Solaris:

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

   - On AIX, HP-UX, and Linux:

     ```
     # 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 autostart 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. To do this, 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 to automate startup for them, you must edit the entries for those instances in the `oratab` file.

---

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

Platform	Initialization File Directory
AIX	`/etc`
Linux and Solaris	`/etc/init.d`
HP-UX	`/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/11.2.0/dbhome_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 $1 ORA_DB
      if [ "$PLATFORM" = "Linux" ] ; then
         touch /var/lock/subsys/dbora
      fi
      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/rc0.d/K01dbora`
HP-UX	`# ln -s /sbin/init.d/dbora /sbin/rc3.d/S990dbora` `# ln -s /sbin/init.d/dbora /sbin/rc0.d/K001dbora`

Linux	`# ln -s /etc/init.d/dbora /etc/rc.d/rc0.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/S99dbora`

Solaris	`# ln -s /etc/init.d/dbora /etc/rc0.d/K01dbora` `# ln -s /etc/init.d/dbora /etc/rc3.d/S99dbora`

3

Configuring Oracle Database

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

- Using Configuration Assistants as Standalone Tools
- Relinking Executables

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

Using Oracle Net Configuration Assistant

When Oracle Net Server or Oracle Net Client is installed, Oracle Universal Installer automatically 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 are multiple databases that you want to upgrade, then you can run Oracle Database Upgrade Assistant after the installation.

If you installed Oracle Database 11*g* 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* 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 the 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 run in the Oracle home directory that you are relinking. In addition, shut down applications linked with Oracle shared libraries.

The `relink` script does not take any arguments.

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

> **See Also:** "Accessing Oracle Database with SQL*Plus" in *Oracle Database Installation Guide for Linux* for more information on how to use the relink script with ASM on Oracle Restart

To relink product executables, run the following command:

```
$ relink
```

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 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
SQL> CONNECT SYSTEM
Enter password: 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. 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.

- Installing SQL*Plus Command-Line Help
- Removing 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. To run this script without interaction, set the `SYSTEM_PASS` environment variable to the SYSTEM user name and password. For example:

– Bourne, Bash, or Korn shell:

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

– C 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
SQL> CONNECT SYSTEM
Enter password: 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
SQL> CONNECT SYSTEM
Enter password: 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. Press **Enter** to return 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 the 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 the 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.

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 Secure Sockets Layer
- 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 the 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 Secure Sockets Layer Protocol Support

The IPC, TCP/IP, and TCP/IP with Secure Sockets Layer 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: *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 the 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 Secure Sockets Layer Protocol Support

The TCP/IP with Secure Sockets Layer protocol support enables an Oracle application on a client to communicate with remote Oracle Database instances through TCP/IP and Secure Sockets Layer. To use TCP/IP with Secure Sockets Layer, you must install Oracle Advanced Security.

The TCP/IP with Secure Sockets Layer 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 Secure Sockets Layer 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 Secure Sockets Layer protocol address:

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

Setting Up the Listener for TCP/IP or TCP/IP with Secure Sockets Layer

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 multiple ports if you intend to start multiple listeners.

If you intend to use Secure Sockets Layer, then you should define a port for TCP/IP with Secure Sockets Layer 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 multiple ports if you intend to start multiple listeners.

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 11*g* Examples media.

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.

This section contains the following topics:

- Precompiler README Files
- Issues Common to All Precompilers
- Static and Dynamic Linking
- Client Shared and Static Libraries

> **Note:** `ORACLE_HOME` in this section refers to `ORACLE_HOME` that is created while installing Oracle Database Client 11*g* by using the Administrator Install type.

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	`pcbcfg.cfg`
Pro*FORTRAN (AIX, HP-UX, and Solaris)	`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 lists the executables for Oracle Precompilers.

Table 6–2 Executables for Oracle Precompilers

Product	Executable
Pro*C/C++	`proc`
Pro*COBOL (AIX, HP-UX, and Solaris SPARC (64-bit))	`procob` or `rtsora`
Pro*COBOL 32-bit (AIX, HP-UX PA-RISC, and Solaris SPARC (64-bit))	`procob32` or `rtsora32`
Pro*FORTRAN (AIX, HP-UX, and Solaris)	`profor` (32-bit for AIX and Solaris)
Pro*FORTRAN 32-bit (HP-UX PA-RISC)	`profor32`
SQL*Module for Ada (AIX)	`modada`

Precompiler README Files

The README files describe changes made to the precompiler since the last release.

Table 6–3 lists the location of the precompiler README files.

Table 6–3 Location of Precompiler README Files

Precompiler	README File
Pro*C/C++	`$ORACLE_HOME/precomp/doc/proc/readme.doc`
Pro*COBOL	`$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 the 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 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: Multiple applications or calls to the same application can use the same dynamic librararies.
- Reduced main memory requirements: The same dynamic library image is loaded into main memory only once, and it can be shared by multiple 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` 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
cannot 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 the 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`	`$ORACLE_HOME/lib32`
HP-UX (64-bit applications) and Linux	`LD_LIBRARY_PATH`	`$ORACLE_HOME/lib`
Solaris (32-bit applications)	`LD_LIBRARY_PATH`	`$ORACLE_HOME/lib32`
Solaris (64-bit applications)	`LD_LIBRARY_PATH_64`	`$ORACLE_HOME/lib`

The client shared library is created automatically during installation. If you must 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 systems.

On HP-UX, 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 the 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	Linux x86
32-bit and 64-bit	AIX, HP-UX PA-RISC, Linux x86-64, and Solaris SPARC
64-bit only	HP-UX Itanium

On AIX, HP-UX, and Solaris SPARC, all demonstrations and client applications provided with Oracle Database 11*g* Release 2 (11.2) 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.

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`
Linux x86-64 and Solaris	`$ORACLE_HOME/lib32/libclntsh.so`	`$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 the platform:

Platform	Environment Variable
AIX	`LIBPATH`
HP-UX (32-bit client applications)	`SHLIB_PATH`
HP-UX, Linux x86, Linux x86-64, and Solaris	`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, and Solaris SPARC 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

This section contains the following topics:

- Pro*C/C++ Demonstration Programs
- Pro*C/C++ User Programs

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 user programs. The following table shows the make files that you can use to build 32-bit and 64-bit user programs with Pro*C/C++:

Platform	64-bit Make File	32-Bit Make File
AIX, HP-UX, Linux x86-64, Solaris x86-64 and Solaris SPARC	`demo_proc.mk`	`demo_proc32.mk`
Linux x86	NA	`demo_proc.mk`

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

Note: On AIX systems, to ensure that the 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

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.

This section contains the following topics:

Pro*COBOL Environment Variables

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

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 HP-UX PA-RISC and Solaris SPARC:

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

- On AIX:

```
ld: rts32: irrecoverable: 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'.
stoped
```

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
```

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:

- 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
```

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, Linux x86-64, and Solaris SPARC	`demo_procob.mk`	`demo_procob_32.mk`
Linux x86	Not applicable	`demo_procob.mk`

> **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 left most column.

Pro*FORTRAN Precompiler

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 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_profor.mk`	`demo_profor_32.mk`

> **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 information about required compiler versions
- *Oracle SQL*Module for Ada Programmer's Guide* for information about SQL*Module for Ada

This section contains the following topics:

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 used to connect 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 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

This section contains the following topics:

- OCI and OCCI Demonstration Programs
- OCI and OCCI User Programs

OCI and OCCI Demonstration Programs

Demonstration programs that show the features of OCI and OCCI are provided with the software through the Oracle Database 11*g* Examples media. 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. Some demonstration programs require specific .sql files to be run, as mentioned in the demonstration source files. OCCI demonstration programs require occidemo.sql to be run.

> **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 build user programs. This make file builds either 32-bit or 64-bit user programs. You can also use the `demo_rdbms32.mk` to build 32-bit user programs on a 64-bit operating system. 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 Linux x86-64	`demo_rdbms.mk`	`demo_rdbms32.mk`
Linux x86	NA	`demo_rdbms.mk`

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

Oracle JDBC/OCI Programs with a 64-Bit Driver

> **Note:**
>
> - The information in this section applies to AIX, HP-UX, Linux x86-64, and Solaris SPARC 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/ojdbc5.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. 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, then 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 linkers are one-pass linkers. The AIX, Linux, and Solaris linkers are two-pass linkers.

- Add the library 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: irrecoverable: 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/11.2.0/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/11.2.0
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, they support 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.

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

Signal	Description
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.
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 multiple signal handles 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 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 the platform) and the client static library (`libclntst11.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 64-Bit Oracle Database PL/SQL

> **Note:** To use the demonstrations described in this chapter, you must install Oracle Database Examples included on the Oracle Database 11*g* Examples media. You must unlock 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 many 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 enter the following command:

   ```
   $ sqlplus
   SQL> CONNECT SCOTT
   Enter password: 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 enter the following command:

```
$ cd $ORACLE_HOME/plsql/demo
$ sqlplus
SQL> CONNECT SCOTT
Enter password: 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, and Solaris:

```
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)
     )
   )
```

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:

```
$ 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
Enter password: system_password
SQL> GRANT CREATE LIBRARY TO SCOTT;
SQL> CONNECT SCOTT
Enter password: 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 64-Bit Oracle Database PL/SQL

Note: This section applies to any 64-Bit Oracle Database.

Starting with Oracle Database 11*g* release 2 (11.2), `extproc32` is no longer available from 64-bit Oracle database install. Therefore, if you have a requirement to run 32-bit external procedures from 64-bit Oracle database, you must obtain 32-bit `extproc` by installing the corresponding 32-bit client software for your platform. Specifically, you must choose custom install within 32-bit client installation, and then select both Oracle Database Utilities and Oracle listener.

In other words, you need a separate Oracle home (32-bit) to run the 32-bit `extproc`. Note that the executable name is not `extproc32` anymore, but simply `extproc`.

To enable 32-bit external procedures on 64-bit Oracle database environment, you must configure 32-bit listener for `extproc` and specify Oracle home (from the 32-bit client install) for the `extproc` `listener.ora` entry.

8

Tuning Oracle Database

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

Importance of Tuning

The intent of this section is to efficiently tune and optimize the performance of Oracle Database. Frequent tuning enhances 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 Input-Output 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

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
```

- On AIX and Linux:

```
$ 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.

On an HP-UX system, the following command displays a summary of Input-Output activity every ten seconds:

```
$ 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 Input-Output activity at a given point in time. If you specify the interval time with multiple options, 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 Input-Output 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 Input-Output requests on a particular disk device must wait to be serviced. Request queues are caused by a high volume of Input-Output requests to that disk or by Input-Output with long average seek times. Ideally, disk request queues should be at or near zero.

swap, swapinfo, swapon, or lsps

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

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

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 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 Base Operation System 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
`nfsstat`	Displays statistics about Network File System and Remote Procedure Call 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 contains tools for monitoring and tuning system activity locally and remotely. The Performance Tool Box consists of two main

components, the Performance Tool Box Manager and the Performance Tool Box Agent. The Performance Tool Box Manager collects and displays data from various systems in the configuration by using the `xmperf` utility. The Performance Tool Box Agent collects and transmits data to the Performance Tool Box Manager by using the `xmserd` daemon. The Performance Tool Box Agent is also available as a separate product called Performance Aide for AIX.

Both Performance Tool Box 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 Input-Output 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 Input-Output, CPU, and memory

usage on a series of screens. You can use the utility to monitor how individual processes are using resources.

- HP Programmer's Analysis Kit

  HP Programmer's Analysis Kit 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

    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 Programmer's Analysis Kit 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 or instruction cycles, along with fast dynamic instrumentation. It is a dynamic performance measurement tool for C, C++, Fortran, and assembly applications.
`gprof`	Creates an execution profile for programs.
`monitor`	Monitors the program counter and calls to certain functions.
`netfmt`	Monitors the network.
`netstat`	Reports statistics on network performance.
`nfsstat`	Displays statistics about Network File System and Remote Procedure Call activity.
`nettl`	Captures network events or packets by logging and tracing.
`prof`	Creates an execution profile of C programs and displays performance statistics for the program, showing where the 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
```

Tuning Memory Management

Start the memory tuning process by measuring paging and swapping space to determine how much memory is available. After you determine the 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 the 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 Memory Resource

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 the 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.

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

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

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

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

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

Use the `vmstat` or `sar` command to monitor paging.

> **See Also:** The man pages or the operating system documentation for information about interpreting the results for the 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 the system consistently has excessive page-out activity, then consider the following solutions:

- Install more memory.
- Move some 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 Input-Output 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.

Allocating Memory Resource

You can set parameters to automatically allocate memory based on the demands of workload and the requirements of various database instances running on the same system. The MEMORY_TARGET parameter specifies the Oracle systemwide usable memory for that instance and automatically tunes SGA and Process Global Area (PGA) components. The MEMORY_MAX_TARGET parameter identifies the value up to which the MEMORY_TARGET parameter can grow dynamically.

By default, the value for both these parameters is zero and there is no auto-tuning. You can activate auto-tuning by setting the MEMORY_TARGET parameter to a nonzero value. To dynamically enable the MEMORY_TARGET parameter, the MEMORY_MAX_TARGET parameter must be set at startup.

Note: If you just set the MEMORY_TARGET parameter to a nonzero value, the MEMORY_MAX_TARGET parameter automatically acquires this value.

The MEMORY_TARGET and MEMORY_MAX_TARGET parameters are only supported on Linux, Solaris, AIX and HP-UX platforms.

On Solaris, Dynamic Intimate Shared Memory is enabled for MEMORY_TARGET or MEMORY_MAX_TARGET.

On Linux, some shared resource requirements are increased when MEMORY_TARGET or MEMORY_MAX_TARGET are enabled. For more information, refer to the "Allocating Shared Resources" section on page A-5.

Tip: You can set the MEMORY_TARGET and MEMORY_MAX_TARGET parameters based on original setup, memory available for Oracle on the computer, and workload memory requirements.

Tuning Disk Input-Output

Balance Input-Output 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.

This section contains the following topics:

- Using Automatic Storage Management
- Choosing the Appropriate File System Type

Using Automatic Storage Management

If you choose to use Automatic Storage Management for database storage, then all database Input-Output is balanced across all available disk devices in the Automatic Storage Management disk group.

By using Automatic Storage Management, you avoid manually tuning disk Input-Output.

Choosing the Appropriate File System Type

Depending on the 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, and Solaris	Unified file system, derived from BSD UNIX
VxFS	AIX, HP-UX, and Solaris	VERITAS file system
ext2/ext3	Linux	Extended file system for Linux
OCFS2	Linux	Oracle cluster file system
JFS/JFS2	AIX	Journaled file system
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. Depending on the file system that you choose, performance differences can be up to 20 percent. 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 Input-Output 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 Other Operating Systems
- Using Disk Resync to Monitor Automatic Storage Management Disk Group

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 Input-Output.

> **Note:** Not all Linux distributions display the `%wio` column in the output of the `sar -u` command. For detailed Input-Output 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 Input-Output subsystem. The higher the sum, the busier the Input-Output 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 Input-Output bound.
- If the `%wio` column value is consistently greater than 20, then the system is Input-Output bound.

Using Disk Resync to Monitor Automatic Storage Management Disk Group

Use the `alter diskgroup disk online` and `alter diskgroup disk offline` commands to temporarily suspend Input-Output to a set of disks. You can use these commands to perform regular maintenance tasks or upgrades such as disk firmware upgrade. If transient failures occur on some disks in a disk group, then use `alter diskgroup disk online` to quickly recover the disk group.

System Global Area

The SGA is the Oracle structure that is located in shared memory. It contains static data structures, locks, and data buffers.

The maximum size of a single shared memory segment is specified by the `shmmax` kernel parameter.

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

Platform	Recommended Value
AIX	NA
HP-UX	The size of the physical memory installed on the system
Linux	Minimum of the following values: ■ Half the size of the physical memory installed on the system ■ 4GB - 1 byte
Solaris	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 the system runs both Oracle9*i* Database and Oracle Database 11*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 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.

Sufficient shared memory must be available to each Oracle process to address the entire SGA:

- Determining the Size of the SGA
- Shared Memory on AIX

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 the database instance, the size of the SGA is displayed next to the Total System Global Area heading.
- 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 Input-Output 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=percent_of_real_memory
```

When using the `maxpin percent` 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 percent=23
```

Note: The default `maxpin percent` value, which is set at 80 `percent`, works for most installations.

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

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.

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.

A

Administering Oracle Database on Linux

This appendix contains information about administering Oracle Database on Linux. From Oracle

It contains the following topics:

- Extended Buffer Cache Support
- Using hugetlbfs on SUSE Linux Enterprise Server 10 or Red Hat Enterprise Linux 4
- Increasing SGA Address Space
- Asynchronous Input-Output Support
- Simultaneous Multithreading
- Allocating Shared Resources
- Database Migration from 32-Bit Linux to 64-Bit Linux

> **Note:** Starting with Oracle Database 11*g* release 2 (11.2), Linux x86-64 media does not contain Linux x86 binaries.You must use Linux x86 media to install 32-bit Oracle home.

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 equals or is 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 Language 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.

Using hugetlbfs on SUSE Linux Enterprise Server 10 or Red Hat Enterprise Linux 4

To enable Oracle Database to use large pages (sometimes called huge pages) on SUSE Linux Enterprise Server 10, or Red Hat Enterprise Linux 4, 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 enough 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).

Increasing SGA Address Space

Note: This section applies to Linux x86 only.

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

- SUSE Linux Enterprise Server 10
- Red Hat Enterprise Linux 4 and Red Hat Enterprise Linux 5

SUSE Linux Enterprise Server 10

To increase the SGA address space on SUSE Linux Enterprise Server 10:

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`:

   ```
   $ sudo sh
   ```

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 4 and Red Hat Enterprise Linux 5

To increase the SGA address space on Red Hat Enterprise Linux 4 and Red Hat Enterprise Linux 5:

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 Input-Output Support

Oracle Database supports kernel asynchronous Input-Output. Asynchronous Input-Output is enabled by default on raw volumes. Automatic Storage Management uses asynchronous Input-Output by default.

By default, the `DISK_ASYNCH_IO` initialization parameter in the parameter file is set to TRUE. To enable asynchronous Input-Output on file system files:

1. Ensure that all Oracle Database files are located on file systems that support asynchronous Input-Output.
2. Set the `FILESYSTEMIO_OPTIONS` initialization parameter in the parameter file to `ASYNCH` or `SETALL`.

Note: If the file system files are managed through ODM library interface or dNFS, asynchronous Input-Output is enabled by default. There is no need to set FILESYSTEMIO_OPTIONS to enable asynchronous Input-Output in these environments.

Simultaneous Multithreading

If Simultaneous Multithreading is enabled, then the `v$osstat` view reports two additional rows corresponding to the online logical (`NUM_LCPUS`) and virtual CPUs (`NUM_VCPUS`).

Allocating Shared Resources

To use the `MEMORY_TARGET` or `MEMORY_MAX_TARGET` feature, the following kernel parameters must be potentially modified.

- `/dev/shm` mount point should be equal in size or larger than the value of `SGA_MAX_SIZE`, if set, or should be set to be at least `MEMORY_TARGET` or `MEMORY_MAX_TARGET`, whichever is larger. For example, with `MEMORY_MAX_TARGET=4GB` only set, to create a 4GB system on the `/dev/shm` mount point:
  - Run the following command as the `root` user:

    ```
    # mount -t tmpfs shmfs -o size=4g /dev/shm
    ```

  - 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=4g 0
    ```

- The number of file descriptors for each Oracle instance are increased by 512*`PROCESSES`. Therefore, the maximum number of file descriptors should be at least this value, plus some more for the operating system requirements. For example, if the `cat /proc/sys/fs/file-max` command returns 32768 and `PROCESSES` are 100, you can set it to 65536 or higher as `root`, to have 51200 available for Oracle. Use one of the following options to set the value for the `file-max` descriptor.
  - Run the following command:

    ```
    echo 65536 > /proc/sys/fs/file-max
    ```

    OR
  - Modify the following entry in the `/etc/sysctl.conf` file and restart the system as `root`.

    ```
    fs.file-max = 65536
    ```

- Per-process number of file descriptors must be at least 512. For example, as `root` run the following command.
  - On bash and sh:

    ```
    # ulimit -n
    ```

  - On csh:

    ```
    # limit descriptors
    ```

If the preceding command returns 200, then run the following command to set the value for the per processor file descriptors limit, for example to 1000:

– On bash and sh:

```
# sudo sh
# ulimit -n 1000
```

– On csh:

```
# sudo sh
# limit descriptors 1000
```

- MEMORY_TARGET and MEMORY_MAX_TARGET cannot be used when LOCK_SGA is enabled. MEMORY_TARGET and MEMORY_MAX_TARGET also cannot be used with huge pages on Linux.

Database Migration from 32-Bit Linux to 64-Bit Linux

To migrate an Oracle Database 11*g* Release 2 (11.2) for 32-bit Linux to an Oracle Database 11*g* Release 2 (11.2) for 64-bit Linux, you must perform the following steps:

- Online Backup of Database With RMAN
- Migrating 32-Bit Linux Database to 64-Bit Linux Database
- Migrating Data To and From ASM

Online Backup of Database With RMAN

Online backup enables to take a backup of the database without having to shutdown the database. To achieve this, perform the following steps:

1. Connect to the database instance as SYSDBA:

```
SQL> CONNECT / AS SYSDBA;
```

2. Run the following commands to ensure that the database is in ARCHIVELOG mode:

```
SQL> SHUTDOWN IMMEDIATE
Database closed
Database dismounted
Oracle instance shutdown

SQL> STARTUP MOUNT
Oracle instance started

Total System Global Area  272629760 bytes
Fixed Size                   788472 bytes
Variable Size             103806984 bytes
Database Buffers          167772160 bytes
Redo Buffers                 262144 bytes
Database mounted

SQL>ALTER DATABASE ARCHIVELOG;
Database altered

SQL> ALTER DATABASE OPEN;
Database altered
```

3. Run the following command to start RMAN, which is located under $ORACLE_HOME/bin directory:

```
RMAN>connect target
```

4. To backup the 32-bit database and all the archived redo log files, run the following command:

```
RMAN>backup database plus archivelog delete input;
```

> **Note:** Archive redo logs are very important to recover the database. Oracle recommends that you back them up along with your database. You can backup the archive redo logs from time to time by issuing the following command:
>
> ```
> RMAN>backup archivelog all delete input;
> ```

Migrating 32-Bit Linux Database to 64-Bit Linux Database

This section covers the following topics:

- Migrating 32-Bit Database to 64-Bit Database With the Same Directory Structure for Data Files
- Migrating 32-Bit Database to 64-Bit Database With Different Directory Structure for Data Files

Migrating 32-Bit Database to 64-Bit Database With the Same Directory Structure for Data Files

If the control file, data file, redo log files have the same structure on the target computer as in the source computer, then perform the following steps:

1. Perform an online backup of the database before starting the migration process. Refer to Online Backup of Database With RMAN on page A-6 for more information.
2. Install Oracle Database 11*g* Release 2 (11.2) for 64-bit Linux in a new Oracle Database home. It is recommended that you use the same version of Oracle Database home as on the 32-bit computer.

   > **See Also:** *Oracle Database Installation Guide for Linux*

3. Copy `init.ora`, data files, control file, and the redo log files from the 32-bit Linux computer to the corresponding locations on the 64-bit Linux computer.
4. Edit `init.ora` file to include the following changes:
   - Update the memory requirements included in the file.
   - Edit the control file location if necessary.

   > **Note:** Oracle recommends that you double the values of shared pool, java pool, `sga_target` and large pool listed in the `init.ora` file.

5. Connect to the database instance as `SYSDBA`:

```
SQL> CONNECT / AS SYSDBA;
```

6. Set the system to spool results to a log file for later verification of success. For example:

```
SQL> SPOOL /tmp/utlirp.log
```

7. Start the 64-bit Oracle Database as follows:

```
SQL> STARTUP UPGRADE pfile=init.ora;
```

> **Note:** *Oracle Database Upgrade Guide* for more information on changing from 32-bit to 64-bite

8. Run the following command on the 64-bit Oracle Database, to invalidate all the PL/SQL modules:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlirp.sql
```

9. Shut down the 64-bit Oracle Database:

```
SQL>shutdown immediate;
```

10. Start the 64-bit Oracle Database:

```
SQL> STARTUP pfile=init.ora;
```

11. Revalidate all the existing PL/SQL modules in the format required by the 64-bit Oracle Database:

```
SQL>$ORACLE_HOME/rdbms/admin/utlrp.sql;
```

Migrating 32-Bit Database to 64-Bit Database With Different Directory Structure for Data Files

If the control file, data file, redo log files have different structure on the target computer as compared to the source computer, then perform the following steps:

1. Perform an online backup of the database before starting the migration process. Refer to Online Backup of Database With RMAN on page A-6 for more information.
2. Install Oracle Database 11*g* Release 2 (11.2) for 64-bit Linux in a new Oracle Database home. It is recommended that you use the same version of Oracle Database home as on the 32-bit computer.

   > **See Also:** *Oracle Database Installation Guide for Linux*

3. Edit `init.ora` file on the 64-bit computer to include the following changes:
   - Update the memory requirements included in the file.
   - The `init.ora` file still contains the 32-bit control file path. You must manually update `control_files` parameter value to include the 64-bit control file location.

   > **Note:** Oracle recommends that you double the values of shared pool, java pool, `sga_target` and large pool listed in the `init.ora` file.

4. If the 64-bit target computer contains a different structure for data files, then you must re-create the control file or mount database on 64-bit computer. Refer to Re-aligning Data File Path and Name on page A-9 for more information.

> **Note:** Oracle recommends not to use the `RESETLOGS` option to re-create control files.

5. Set the system to spool results to a log file for later verification of success. For example:

```
SQL> SPOOL /tmp/utlirp.log
```

6. Run the following command on the 64-bit Oracle Database, to invalidate all the PL/SQL modules:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlirp.sql
```

7. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF;
```

8. Shut down the 64-bit Oracle Database:

```
SQL>shutdown immediate;
```

9. Start the 64-bit Oracle Database:

```
SQL> STARTUP pfile=init.ora;
```

10. Revalidate all the existing PL/SQL modules in the format required by the 64-bit Oracle Database:

```
SQL>$ORACLE_HOME/rdbms/admin/utlrp.sql;
```

Re-aligning Data File Path and Name

The following are some methods to realign the data file names and path to point to the correct location:

- Re-creating Control File
- Mounting Database on a 64-Bit Computer

Re-creating Control File

Perform the following steps to re-create the control file:

1. Run the following command to backup the control file to trace. The trace file is located under the `diagnostic_dest` directory on the 32-bit Linux computer. The following command generates a trace file which contains the necessary sql to re-create the control file:

```
SQL> alter database backup controlfile to trace;
```

> **Note:** Ensure that you open the Oracle Database in the `UPGRADE` mode after the control file is created.

2. Rename the trace file generated into `.sql` format on the 32-bit Linux computer. For example:

```
$ cp trace.ora control.sql
```

   The contents of the control file are as follows, for example:

- Re-creating control files with NORESETLOGS option.

  32-bit control file with NORESETLOGS option:

```
STARTUP NOMOUNT pfile=t_init1.ora
CREATE CONTROLFILE REUSE DATABASE "L32" NORESETLOGS  NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 454
LOGFILE
     GROUP 1 '/ade/aime_l32/oracle/dbs/t_log1.f'  SIZE 25M,
     GROUP 2 '/ade/aime_l32/oracle/dbs/t_log2.f'  SIZE 25M
DATAFILE
     '/ade/aime_l32/oracle/dbs/t_db1.f'
     '/ade/aime_l32/oracle/dbs/t_ax1.f'
     '/ade/aime_l32/oracle/dbs/t_undo1.f'
CHARACTER SET WE8DEC;
RECOVER DATABASE;
ALTER DATABASE OPEN UPGRADE;
ALTER TABLESPACE TEMP ADD TEMPFILE '/ade/aime_l32/oracle/dbs/t_tmp1.f' SIZE
41943040  REUSE AUTOEXTEND ON NEXT 8192  MAXSIZE 32767M;
```

  Now, let us consider the modified 64-bit control file:

```
STARTUP NOMOUNT pfile=t_init1.ora
CREATE CONTROLFILE REUSE DATABASE "L32" NORESETLOGS  NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 454
LOGFILE
     GROUP 1 '/ade/aime_l64/oracle/dbs/t_log1.f'  SIZE 25M,
     GROUP 2 '/ade/aime_l64/oracle/dbs/t_log2.f'  SIZE 25M
DATAFILE
     '/ade/aime_l64/oracle/dbs/t_db1.f'
     '/ade/aime_l64/oracle/dbs/t_ax1.f'
     '/ade/aime_l64/oracle/dbs/t_undo1.f'
CHARACTER SET WE8DEC;
RECOVER DATABASE;
ALTER DATABASE OPEN UPGRADE;
ALTER TABLESPACE TEMP ADD TEMPFILE '/ade/aime_l64/oracle/dbs/t_tmp1.f' SIZE
41943040  REUSE AUTOEXTEND ON NEXT 8192  MAXSIZE 32767M;
```

- Re-creating control files with RESETLOGS option:

  32-bit control file with RESETLOGS option:

```
STARTUP NOMOUNT pfile=t_init1.ora
CREATE CONTROLFILE REUSE DATABASE "L32" RESETLOGS  NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 454
LOGFILE
     GROUP 1 '/ade/aime_l32/oracle/dbs/t_log1.f'  SIZE 25M,
     GROUP 2 '/ade/aime_l32/oracle/dbs/t_log2.f'  SIZE 25M
DATAFILE
     '/ade/aime_l32/oracle/dbs/t_db1.f'
```

```
     '/ade/aime_l32/oracle/dbs/t_ax1.f'
     '/ade/aime_l32/oracle/dbs/t_undo1.f'
CHARACTER SET WE8DEC;
RECOVER DATABASE USING BACKUP CONTROLFILE;
ALTER DATABASE OPEN RESETLOGS UPGRADE;
ALTER TABLESPACE TEMP ADD TEMPFILE '/ade/aime_l32/oracle/dbs/t_tmp1.f' SIZE
41943040  REUSE AUTOEXTEND ON NEXT 8192  MAXSIZE 32767M;
```

Now, let us consider the modified 64-bit control file:

```
STARTUP NOMOUNT pfile=t_init1.ora
CREATE CONTROLFILE REUSE DATABASE "L32" RESETLOGS  NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 454
LOGFILE
     GROUP 1 '/ade/aime_l64/oracle/dbs/t_log1.f'  SIZE 25M,
     GROUP 2 '/ade/aime_l64/oracle/dbs/t_log2.f'  SIZE 25M
DATAFILE
     '/ade/aime_l64/oracle/dbs/t_db1.f'
     '/ade/aime_l64/oracle/dbs/t_ax1.f'
     '/ade/aime_l64/oracle/dbs/t_undo1.f'
CHARACTER SET WE8DEC;
RECOVER DATABASE USING BACKUP CONTROLFILE;
ALTER DATABASE OPEN RESETLOGS UPGRADE;
ALTER TABLESPACE TEMP ADD TEMPFILE '/ade/aime_l64/oracle/dbs/t_tmp1.f' SIZE
41943040  REUSE AUTOEXTEND ON NEXT 8192  MAXSIZE 32767M;
```

3. Based on the method employed to realign the file paths to point to the correct locations, you must copy the necessary files from the source 32-bit Linux computer to the target 64-bit Linux computer:
   - NORESETLOGS option: Copy `init.ora`, data files, re-created control files (*`control.sql`*), and the redo log files from the 32-bit Linux computer to the corresponding locations on the 64-bit Linux computer
   - RESETLOGS option: Copy `init.ora`, data files, and re-created control files (*`control.sql`*) from the 32-bit Linux computer to the corresponding locations on the 64-bit Linux computer
4. Connect to the database instance as `SYSDBA`:

   ```
   SQL> CONNECT / AS SYSDBA;
   ```

5. To change from 32-bit to 64-bit, run the following command from the Linux 64-bit Oracle Database home:

   ```
   sql>set echo on
   sql>@control.sql
   ```

Mounting Database on a 64-Bit Computer

Perform the following steps to mount the database on a 64-bit computer:

1. Copy `init.ora`, data files, control file and the redo log files from the 32-bit Linux computer to the corresponding locations on the 64-bit Linux computer
2. Connect to the database instance as `SYSDBA`:

   ```
   SQL> CONNECT / AS SYSDBA;
   ```

3. Start the 64-bit Oracle Database as follows:

```
SQL> STARTUP mount pfile=init.ora;
```

4. Update all the 32-bit data file locations to include the 64-bit data file locations. For example:

```
sql> alter database rename file '/ade/aime_l32/oracle/dbs/t_db1.f' to
'/ade/aime_l64/oracle/dbs/t_db1.f'
sql> Database altered.
```

5. Update all the 32-bit log file locations to include the 64-bit log file locations. For example:

```
sql> alter database rename file '/ade/aime_l32/oracle/dbs/t_log.f' to
'/ade/aime_l64/oracle/dbs/t_log.f'
sql> Database altered.
```

6. To change from 32-bit to 64-bit, run the following command from the Linux 64-bit Oracle Database home:

```
sql> ALTER DATABASE OPEN UPGRADE;
```

Note: Refer to *Oracle Database Upgrade Guide* for more information on changing from 32-bit to 64-bit

Migrating Data To and From ASM

To take advantage of Automatic Storage Management (ASM), you can migrate an Oracle 11*g* Release 2 (11.2) database into and out of an ASM disk group using Recovery Manager (RMAN). This migration is performed using RMAN even if you are not using RMAN for your primary backup and recovery strategy.

See Also: Chapter 8, "Performing ASM Data Migration with RMAN", in *Oracle Database Storage Administrator's Guide* for more information on migrating databases.

B

Using Oracle ODBC Driver

Note: Oracle ODBC Driver is certified on all the UNIX platforms for 11.2.

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 B-23 for information about the DATE and TIMESTAMP data types related to performance and tuning

Floating Point Data Types

When connected to an Oracle Database 11*g* Release 2 (11.2) or later, 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.

Limited Data Type	Description
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.
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.

Keyword	Meaning	Value
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.
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 the 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 the 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
- Implementation of ODBC API Functions
- Implementation of the ODBC SQL Syntax
- Implementation of Data Types

API Conformance

Oracle ODBC Driver release 10.2.0.1.0 and higher supports all Core, Level 2, and Level 1 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, then 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 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 must explicitly call entry points with the suffix. An ODBC application that is compiled with the `UNICODE` and `_UNICODE` preprocessor definitions generates the appropriate calls. For example, a call to `SQLPrepare` compiles 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 must 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 must 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, then 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, then there would be an additional conversion to revert the data 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 B–1 Connection to Database

To use this code, you only must 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 B–2 Simple Retrieval

The following example retrieves the employee names and the job tiles from the EMP table. With the exception that you must 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 must 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 B–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 B–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 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 B–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 frequently connects and disconnects from a data source. Reusing pooled connections is extremely efficient compared to reestablishing a connection.
- Minimize the number of times a statement must 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 does 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.

See Also: *Oracle Database SecureFiles and Large Objects Developer's Guide* for more information on LOBs

---

Note: LOB data compression enables you to compress SecureFiles to gain disk, Input-Output, and redo logging savings. This reduces costs as compression utilizes space most efficiently and improves the performance of SecureFiles as compression reduces Input-Output and redo logging.

LOB data encryption provides enhanced database security. While the encrypted data is available for random reads and writes, the data is more secure.

Data compression and encryption consumes some additional memory.

- 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 reuses 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 determines how many rows of data Oracle ODBC Driver pre-fetches 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 is 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 the 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 B-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, then 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 `ORA-` prefix, then you can obtain more information about the error in Oracle Database documentation.

C

Database Limits

This appendix describes database limits.

Database Limits

Table C–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 C–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	1055

Table C–2 lists the Oracle Database file size limits in bytes.

Table C–2 File Size Limits

File Type	Platform	File size limit
Data files	Any	4,194,303 multiplied by the value of the `DB_BLOCK_SIZE` parameter
Import/Export files and SQL*Loader files	AIX, HP-UX, Linux, and Solaris: 32-bit with 32-bit files	2,147,483,647 bytes
	AIX, HP-UX, Linux, and Solaris: 64-bit files	Unlimited **Note**: File size is limited by limits allowed by the operating system on different file systems.
Control files	HP-UX, Linux, and Solaris	20000 database blocks
	AIX	10000 database blocks

D

Managing Input Output Resources

This appendix explains improvements in the Oracle Database Resource Manager for Oracle Database 10*g* series. It includes information about the following topics:

Overview

In Oracle Database 10*g*, Oracle Database Resource Manager manages the CPU resource among Oracle processes in an instance. However, at times those processes are bound by the input-output. It would be more effective to be able to also manage the input-output bandwidth. The new Input Output Resource Manager feature helps manage dynamic resources involved in handling disk input-output operations.

You can enable Input Output Resource Manager on all platforms on which Oracle runs. It works on small and large database configurations, and on hardware configurations. Further, Input Output Resource Manager scales with both, the number of disks and the additional channel capacity.

Input Output Resource Manager permits you to perform the following tasks:

- You can enable or disable Input Output Resource Manager without restarting the instance.
- You can disable Input Output Resource Manager independent of CPU resource management.
- You can load a new input-output resource plan on a running system.
- You can change an existing plan on a running system.
- You can switch plans and expect a smooth change in the input-output bandwidth distribution.

Enabling Input Output Resource Manager does not cause any significant decrease in performance.

- At times, honoring resource allocations can result in sub-optimal disk utilization. In such cases, there is an expected decrease in throughput.
- If the input-output load is under the database's limits, then enabling Input Output Resource Manager with a plan with a single consumer group has a minimal effect on performance.

- For consumer groups with high resource allocations, using Input Output Resource Manager for plans with multiple consumer groups shows a significant performance improvement.

Requirements

The following is the list of the requirements for Input Output Resource Manager:

- You must specify the type of database configuration, whether it is a dedicated or shared storage configuration, in the resource manager plan.

  Note:

  Specify the database type for each database.

  The storage configuration applies to the database as a whole, not a specific plan.

- If a database configuration is a shared storage configuration, then you must do the following:
  - Specify a per database cap on the maximum amount of input-output bandwidth that the database can use. In this way, you can set a limit to the input-output from one database, when different databases use the same storage.
  - Configure the MBPS and IOPS input-output limits in the plan.
- After enabling Input Output Resource Manager, you must specify the same plan for each instance. This is because the resource manager plan specifies the type of storage configuration and the IOPS and MBPS caps if the storage is a shared storage configuration.
- You must specify which database files must be managed as part of the main, critical storage pool.
- You must specify which database files must be managed by configuring rules based on the files' Automatic Storage Management disk group, file type, and area.
- If you partitioned the input-output resources between Oracle and another application using an external OS-level Input Output Resource Manager, then you need do the following:
  - Specify input-output caps for the Oracle database.
  - Use Input Output Resource Manager for intra-database resource management.
- You must categorize all database files into any one of the following categories:
  - Managed files
  - Unmanaged files
- You can specify which files are not managed. For example, a temp file, a log file, or a file in the recovery area.

> **Note:** A file is not managed if it is from an excluded Automatic Storage Management disk group. In general, only one Automatic Storage Management disk group is managed, as Automatic Storage Management disk groups are typically on separate storage devices. Multiple Automatic Storage Management disk groups may be managed, if they share storage controllers or disks.

The rules for unmanaged files apply to the whole database, and not just a particular plan.

- You can specify the following for each plan and for each group:
  - The maximum amount of input-output and input-output requests that a session can issue before an action is taken, stop session, stop call, or switch to another group.
  - The maximum number of input-output requests that a session can issue as a batch.
- For dedicated storage configurations, you must perform an input-output calibration, either while creating the database, or later, by using a PL/SQL command. Ensure that you run the input-output calibration tool before turning on Input Output Resource Manager for the first time.

> **Note:** Calibration requires the issuance of sufficient amount of input-output to saturate the storage system. This can affect the performance of critical sessions. Therefore, you must calibrate input-output when the database is inactive. It takes approximately 10 minutes. Only an administrator with SYSDBA privilege can run this procedure.

If you are unable to run an input-output calibration tool, then the available input-output resources are estimated based on the number of host devices, operating system statistics, and Oracle statistics. You must provide an initial estimate of the input-output limits. Input Output Resource Manager is not enabled until an hour of input-output statistics is available.

PL/SQL Statement

You can calibrate the input-output capabilities of the storage, by using the following PL/SQL statement:

```
DBMS_RESOURCE_MANAGER.CALIBRATE_IO()
```

Input-output calibration involves obtaining the storage's capabilities by issuing an extremely heavy input-output workload.

Storage capability is calculated in terms of how many input-output requests and bytes of input-output can be sustained for each second.

Input-output workload consists of random reads. It uses single database block reads and large reads whose size is determined by the maximum size of input-output that is supported for the host's operating system.

Note: If you excluded any files from being managed, then the unmanaged files are not used for input-output calibration.

Status and Results View

The input-output calibration procedure returns immediately, however, the results are not available immediately. You must refer to the `V$IO_CALIBRATION` table for the status and results of input-output calibration.

Table D–1 lists the components of a `V$IO_CALIBRATION` table.

Table D–1 Components of a V$IO_CALIBRATION Table

Component	Description
STATUS	Provides the status of input-output calibration, whether it's in progress or ready. Results are available when the status changes to ready.
START_TIME	Provides the time at which the input-output calibration was initiated.
END_TIME	Provides the time at which the input-output calibration completed.
MAX_IOPS	Provides the maximum number of read requests that can be sustained for each second. The size of the read is specified by the parameter DB_BLOCK_SIZE. The reads are issued so that they are as randomly scattered as possible.
MAX_MBPS	Provides the maximum number of bytes of reads that can be sustained for each second. The size of the read is the maximum possible for the host's operating system platform, which is 1 MB. The reads are issued so that they are as randomly scattered as possible.

E

Very Large Memory on Linux x86

Oracle Database for Linux 32-bit supports Very Large Memory (VLM) configurations, which allows Oracle Database to access more than the 4 GB of RAM traditionally available to Linux applications. The Oracle Very Large Memory mode is used to create a large database buffer cache. Running the computer in Very Large Memory mode requires setup changes to Linux and imposes some limitations on the features, and `init.ora` parameters.

This appendix includes the following topics:

- Shared Global Area Tuning
- Overview of HugePages
- Methods To Increase SGA Limits
- Configuring Very Large Memory for Oracle Database
- Restrictions Involved in Implementing Very Large Memory

Shared Global Area Tuning

The basic memory structures associated with Oracle Database include System Global Area (SGA) and Program Global Area (PGA). The SGA is a group of shared memory structures, known as SGA components, that contain data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. The PGA is a memory region that contains data and control information for a server process.

The following topics are covered in this section:

- Manual SGA Tuning
- Automatic SGA Tuning

Manual SGA Tuning

To increase the System Global Area limit on a 32-bit computer, you must manually set the initialization parameters `DB_BLOCK_BUFFERS` and `SHARED_POOL_SIZE` to values you have chosen for Oracle Database. The initialization parameter `DB_BLOCK_SIZE` sets the block size and the buffer cache size for an instance.

A typical 32-bit Linux kernel splits 4 GB address space into 3 GB of user space and 1 GB of kernel. For example, let us consider a configuration with 3 GB user space and 1 GB kernel. On normal kernel, the following is the default memory layout for every process:

1 GB text code

1.7 GB available process memory for address space

0.3 GB stack

1.0 GB kernel

From the preceding example, it is clear that 1.7 GB is left for SGA. This can be effectively used by the shared pool.

On the 1 GB kernel addressable space, the following memory zones are available:

- ZONE_DMA
- ZONE_NORMAL
- ZONE_HIMEM

ZONE_DMA

This zone is mapped to the first 16 MB of the physical memory. Some devices use this zone for data transfer.

ZONE_NORMAL

This zone is mapped from 16 MB to 896 MB of the physical memory. This zone is also known as the low memory zone. It is permanently mapped to the kernel space. This is a performance-critical zone as most of the kernel resources reside in this space and can operate from here. Thus, running kernel resource intensive applications can put pressure in the low memory. Low memory starvation can result in performance degradation or even system hang.

ZONE_HIMEM

Linux cannot access memory that is not mapped directly into its address space. The kernel maps the physical pages to the virtual address space before it can use memory greater than 1 GB. To access the pages in the ZONE_HIGHMEM zone, the kernel maps these pages in the ZONE_NORMAL zone.

Automatic SGA Tuning

Starting with 10*g*, `SGA_TARGET` parameter is introduced to facilitate automatic tuning of database parameters such as shared pool, java pool, large pool, and database buffer cache. You must set the value of `SGA_TARGET` parameter to a non zero value to grow and shrink auto-tuned memory components.

With 11*g*, this feature was reviewed and a new parameter called `MEMORY_TARGET` was introduced which facilitates auto tuning of SGA and PGA components of the memory.

Dynamic SGA and multiple block size are not supported with Very Large Memory. To enable Very Large Memory on the system, you must ensure that you set the value of `SGA_TARGET` or `MEMORY_TARGET` to zero. If Very Large Memory is enabled on the system, then the Database buffer cache can use extra memory.

Note: Oracle recommends that you unset `SGA_TARGET` or `MEMORY_TARGET` parameter before implementing Very Large Memory on the same system.

Overview of HugePages

HugePages is a feature integrated into the Linux kernel with release 2.6. It is a method to have larger pages where it is useful for working with very large memory. It can be useful for both 32-bit and 64-bit configurations. HugePage sizes vary from 2MB to 256MB, depending on the kernel version and the hardware architecture. For Oracle Databases, using HugePages reduces the operating system maintenance of page states, and increases TLB (Translation Lookaside Buffer) hit ratio.

Without HugePages, the operating system keeps each 4 KB of memory as a page, and when it is allocated to the SGA, then the lifecycle of that page (dirty, free, mapped to a process, and so on) must be kept up to date by the operating system kernel.

With HugePages, the operating system page table (virtual memory to physical memory mapping) is smaller, since each page table entry is pointing to pages from 2 MB to 256 MB. Also, the kernel has fewer pages whose lifecyle must be monitored.

For example, if you use HugePages with 64-bit hardware, and you want to map 256 MB of memory, you may need one page table entry (PTE). If you do not use HugePages, and you want to map 256 MB of memory, then you must have 256 MB * 1024 KB/4 KB = 65536 PTEs.

Methods To Increase SGA Limits

The following are the methods to increase SGA limits on a 32-bit computer:

- Hugemem Kernel
- Hugemem Kernel with Very Large Memory

Hugemem Kernel

To use large memory system, Linux `hugemem` kernel can be employed. This enables creating SGA of up to 3.6 GB. The `hugemem` kernel allows 4GB/4GB split of the kernel and user space address. A 32-bit computer splits the available 4 GB address space into 3 GB virtual memory space that can be used to run user-defined processes and a 1 GB space for the kernel. However, for a larger SGA, you must use `hugemem` kernel.

The `hugemem` kernel provides support for accessing memory beyond 4 GB virtual addressable limit of 32-bit kernel to access 16 GB, and up to 64 GB of physical memory. The `hugemem` kernel on large computers ensures better stability as compared to the performance overhead of address space switching. This kernel supports a 4 GB per process user space (as against 3 GB for the other kernels) and a 4 GB direct kernel space. The `hugemem` kernel is used when you are installing the software on a multiprocessor computer with more than 4 GB of main memory. It is also useful for running configurations with less memory.

Run the following command to determine if you are using the `hugemem` kernel:

```
$ uname -r
2.6.9-5.0.3.ELhugemem
```

For example, let us consider a configuration with 4 GB user space and 4 GB kernel. If you reduce the SGA attached, then it results in 3.42 GB process memory as follows:

3.42 GB available process memory for address space

0.25 GB kernel "trampoline"

0.33 GB SGA base

4.0 GB kernel

Hugemem Kernel with Very Large Memory

On a 32-bit Physical Address Extension (PAE) system with `hugemem`, Oracle database can use a shared memory file system, feature called Very Large Memory (VLM). Very Large Memory can increase SGA from 1.7 GB to 62 GB. However, the user space address is still limited to 4 GB.

You must enable the Very Large Memory feature for Oracle to use a shared memory file system. This feature moves the buffer cache of the SGA from the System V shared memory to the shared memory file system. For Red Hat Enterprise Linux 4.0 / Oracle Enterprise Linux 4.0 to use the Very Large Memory option to create a very large buffer cache, you have two options:

- Use `shmfs`: mount a `shmfs` with a certain size to `/dev/shm`, and set the correct permissions. In Red Hat Enterprise Linux 4.0 / Oracle Enterprise Linux 4.0, `shmfs` allocated memory is pageable.
- Use `ramfs`: `ramfs` is similar to `shmfs`, except that pages are not pageable or swappable. This approach provides the commonly desired effect. `Ramfs` is created by `mount -t ramfs ramfs /dev/shm` (`unmount /dev/shm` first). The only difference here is that the `ramfs` pages are not backed by big pages.

> **Note:** `USE_INDIRECT_DATA_BUFFERS=TRUE` must be present in the initialization parameter file for the database instance that use Very Large Memory support. If this parameter is not set, then Oracle Database 11*g* Release 2 (11.2) or later behaves in exactly the same way as previous releases.

You can increase the SGA to about 62 GB on a 32-bit computer with 64 GB RAM. Very Large Memory configurations improve database performance by caching more database buffers in memory. This reduces disk I/O compared to configurations without Very Large Memory. The Page Address Extension (PAE) feature of the processor enables physical addressing of 64 GB of RAM. However, you cannot attach a program to shared memory directly if it is 4 GB or more in size because PAE does not enable a program to an address, if it is more than 4 GB, directly. To resolve this issue, you can create a shared memory file system that can be as large as the virtual memory supported by the kernel. You can use this shared file system for a program to dynamically attach to the regions of this file system. A shared memory file system enables large applications, such as Oracle, to access a large shared memory on a 32-bit computer.

Very Large Memory uses 512 MB of the non-buffer cache SGA to manage itself. Very Large Memory uses this memory area to map the indirect data buffers, such as shared memory file system buffer, into the process address space.

For example, if you have 2.5 GB of non-buffer cache SGA for shared pool, then only 2 GB of non-buffer cache SGA is available for Shared pool, Large pool, and Redo Log buffer. Remaining 512 MB of the non-buffered cache is used to manage Very Large Memory.

Configuring Very Large Memory for Oracle Database

Complete the following procedure to configure Very Large Memory on the computer:

1. Log in as a `root` user:

```
sudo -sh
Password:
```

2. Edit the `/etc/rc.local` file and add the following entries to it to configure the computer to mount `ramfs` over the `/dev/shm` directory, whenever you start the computer:

```
umount /dev/shm
mount -t ramfs ramfs /dev/shm
chown oracle:oinstall /dev/shm
```

   In the preceding commands, `oracle` is the owner of Oracle software files and `oinstall` is the group for Oracle owner account.

3. Restart the server.
4. Log in as a `root` user.
5. Run the following command to check if the `/dev/shm` directory is mounted with the ramfs type:

```
# mount | grep shm
ramfs on /dev/shm type ramfs (rw)
```

6. Run the following command to check the permissions on the `/dev/shm` directory:

```
# ls -ld /dev/shm
drwxr-xr-x  3 oracle oinstall 0 Jan 13 12:12 /dev/shm
```

7. Edit the `/etc/security/limits.conf` file and add the following entries to it to increase the max locked memory limit:

```
soft    memlock        3145728
hard    memlock        3145728
```

8. Switch to the `oracle` user:

```
# sudo - oracle
Password:
```

9. Run the following command to check the max locked memory limit:

```
$ ulimit -l
3145728
```

10. Complete the following procedure to configure instance parameters for Very Large Memory:
    a. Replace the `DB_CACHE_SIZE` and `DB_xK_CACHE_SIZE` parameters with `DB_BLOCK_BUFFERS` parameter.
    b. Add the `USE_INDIRECT_DATA_BUFFERS=TRUE` parameter.
    c. Configure SGA size according to the SGA requirements.
    d. Remove SGA_TARGET, if set.
11. Start the database instance.
12. Run the following commands to check the memory allocation:

```
$ ls -l /dev/shm
$ ipcs -m
```

13. Run the following command to display the value of `Hugepagesize` variable:

```
$ grep Hugepagesize /proc/meminfo
```

14. Complete the following procedure to create a script that computes recommended values for `hugepages` configuration for the current shared memory segments:

    a. Create a text file named `hugepages_settings.sh`.

    b. Add the following content in the file:

```
#!/bin/bash
#
# hugepages_settings.sh
#
# Linux bash script to compute values for the
# recommended HugePages/HugeTLB configuration
#
# Note: This script does calculation for all shared memory
# segments available when the script is run, no matter it
# is an Oracle RDBMS shared memory segment or not.
# Check for the kernel version
KERN='uname -r | awk -F. '{ printf("%d.%d\n",$1,$2); }''
# Find out the HugePage size
HPG_SZ=`grep Hugepagesize /proc/meminfo | awk {'print $2'}`
# Start from 1 pages to be on the safe side and guarantee 1 free HugePage
NUM_PG=1
# Cumulative number of pages required to handle the running shared memory
segments
for SEG_BYTES in `ipcs -m | awk {'print $5'} | grep "[0-9][0-9]*"`
do
   MIN_PG='echo "$SEG_BYTES/($HPG_SZ*1024)" | bc -q'
   if [ $MIN_PG -gt 0 ]; then
      NUM_PG=`echo "$NUM_PG+$MIN_PG+1" | bc -q`
   fi
done
# Finish with results
case $KERN in
   '2.4') HUGETLB_POOL=`echo "$NUM_PG*$HPG_SZ/1024" | bc -q`;
          echo "Recommended setting: vm.hugetlb_pool = $HUGETLB_POOL" ;;
   '2.6') echo "Recommended setting: vm.nr_hugepages = $NUM_PG" ;;
    *) echo "Unrecognized kernel version $KERN. Exiting." ;;
esac
# End
```

    c. Run the following command to change the permission of the file:

```
$ chmod +x hugepages_settings.sh
```

15. Run the hugepages_settings.sh script to compute the values for `hugepages` configuration:

```
$ ./hugepages_settings.sh
```

> **Note:** Before running this script, ensure that all the applications that must use `hugepages` run.

16. Set the following kernel parameter:

```
# sysctl -w vm.nr_hugepages=value_displayed_in_step_15
```

17. To make the value of the parameter available for every time you restart the computer, edit the `/etc/sysctl.conf` file and add the following entry:

    ```
    vm.nr_hugepages=value_displayed_in_step_15
    ```

18. Run the following command to check the available `hugepages`:

    ```
    $ grep Huge /proc/meminfo
    ```

19. Restart the instance.

20. Run the following command to check the available `hugepages` (1 or 2 pages free):

    ```
    $ grep Huge /proc/meminfo
    ```

> **Note:** If the setting of the `nr_hugepages` parameter is not effective, you might must restart the server.

Restrictions Involved in Implementing Very Large Memory

Following are the limitations of running a computer in the Very Large Memory mode:

- Very Large Memory configurations do not support multiple database block sizes.
- The computer on which Oracle Database is installed must have more than 4 GB of memory.
- Initialization parameters `SGA_TARGET` or `MEMORY_TARGET` must be unset before you implement Very Large Memory.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V

X