Oracle8™

Administrator's Reference

Release 8.0.6 (32 Bit) for SGI® IRIX™

March 2000

Part No. Z26669-01

ORACLE®

Oracle8 Administrator's Reference, Release 8.0.6 (32 Bit) for SGI® IRIX™

Part No. Z26669-01

Contents

3 Tuning Oracle8 on SGI IRIX

4 Administering SQL*Plus on SGI IRIX Systems

5 Using Oracle Precompilers and the Oracle Call Interface

6 Configuring Oracle Net8

7 Running Oracle Data Cartridge Demonstrations on SGI IRIX

Index

Send Us Your Comments

Oracle8 Administrator's Reference (32 Bit) for SGI IRIX, Release 8.0.6

Part No. Z26669-01

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

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

If you find any errors or have suggestions for improvement, please indicate the book title, part number, chapter, and section. You can send comments to us in the following ways:

Email: writers@ie.oracle.com

Fax: +353 1 8033 321 Attn: Technical Publications Manager

Mail: Technical Publications Manager
Oracle Corporation
European Development and Technology Centre
Temple Rd.
Blackrock, Co. Dublin
Ireland

If you would like a reply, please give your name, postal or email address, and telephone number.

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

Preface

This guide supplements generic server administration information found in the *Oracle8 Administrator's Guide* and the *Oracle8 Tuning Guide*. This preface contains the following sections:

- Purpose
- Audience
- Oracle8 and Oracle8 Enterprise Edition
- Typographic Conventions
- Command Syntax
- Contacting Customer Support
- Related Documentation

Purpose

This reference provides information specific to the IRIX operating system that is required to successfully administer and tune Oracle8. This reference supplements information found in the documentation set for Oracle8 and Oracle8 Enterprise Edition.

Audience

This document is intended for anyone responsible for administering Oracle8 on an SGI IRIX system.

Oracle8 and Oracle8 Enterprise Edition

Unless noted otherwise, features and functionality described in this document are common to both Oracle8 and Oracle8 Enterprise Edition. For more information about the features and functionality of Oracle8 Enterprise Edition, refer to the *Oracle8 Release Notes (32 Bit) for SGI IRIX*.

Typographic Conventions

The following conventions apply to text in this guide:

`monospace`	Monospace type indicates UNIX commands, directory names, path names, and file names.
brackets [ ]	Words enclosed in brackets indicate key names (for example, Press [Return]). Note that brackets have a different meaning when used in command syntax.
italics	Italic type indicates a variable, including variable portions of file names, or emphasis.
UPPERCASE	Uppercase letters indicate Structured Query Language (SQL) commands, initialization parameters, or environment variables.

Because UNIX is case-sensitive, conventions in this document might differ from those used in Oracle product documentation.

Command Syntax

Command syntax appears in `monospace` font and assumes the use of the Bourne shell. The dollar prompt ($) at the beginning of the UNIX command examples is the default UNIX command prompt. Do not enter it in the examples. The following conventions apply to command syntax:

backslash \	A backslash indicates a command that is too long to fit on a single line. Enter the line as printed (with a backslash) or enter it as 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`*`]` Note that brackets have a different meaning when used in regular text.
ellipses ...	Ellipses indicate an arbitrary number of similar items: `CHKVAL fieldname` *`value1 value2 ... valueN`*
italics	Italic type indicates a variable. Substitute a value for the variable: *`library_name`*
vertical line |	A vertical line indicates a choice within braces or brackets: *`SIZE filesize [K|M]`*

Contacting Customer Support

To contact Oracle Worldwide Customer Support (WWCS), call the number listed for the nearest support center to your time zone. Note that the hours are specified in your support contract:

Location of Support Centre	Phone Number
United States	+1.650.506.1500
Europe	+44.1344.860160
Asia	+81.3.5717.1850

Please prepare the following information before you call:

- ❑ Your CSI number (if applicable) or complete contact details, including any special project information relating to the problem
- ❑ The release levels of the Oracle Server and associated products (for example, Oracle8 Server release 8.0.6.0, and Oracle Forms release 4.5.6.3.2)
- ❑ Operating system name and release level, including patches and packages
- ❑ Details of error codes, numbers, and descriptions associated with the problem
- ❑ A full description of the problem, providing answers to the following questions:
  - What happened? For example, the command used and result obtained.
  - When did it happen? For example, time of day, or after a particular command, or after an operating system or Oracle upgrade.
  - Where did it happen? For example, on a particular system, or within a particular procedure or table.
  - What is the extent of the problem? For example, is your production system unavailable, or is the impact less severe? Is the problem getting worse?
  - Is there anything you expected to happen that did not?
- ❑ Copies of any trace files, core dumps, or log files recorded when the problem occurred

For installation-related problems, please provide the following information:

- ❑ Listings of the contents of the `$ORACLE_HOME` directory, and any staging area, if applicable
- ❑ Contents of the installation log files in the `$ORACLE_HOME/orainst` directory: `install.log`, `sql.log`, `make.log`, and `os.log`

Note: For more information about Worldwide Customer Support Services, refer to the web page at the following URL:

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

Related Documentation

This document describes how to administer a basic configuration of Oracle8. Advanced configuration and tuning information for production database systems is provided in the following guides:

- *Oracle8 Administrator's Guide.* Use this guide as a starting point for tasks associated with the Oracle8 Server, such as database creation, managing database objects, and creating users.
- *Net8 Administrator's Guide*
- *Oracle8 Tuning*

For more information specific to running Oracle8 on SGI IRIX systems, refer to the following guides:

- *Oracle8 Installation Guide (32 Bit) for SGI IRIX*.

If you are unfamiliar with the concepts or terminology associated with relational database management systems, read Chapter 1 in the *Oracle8 Concepts* guide before beginning your installation.

Ordering Documentation

To order documentation:

- In the United States, call Documentation Sales at: **1.800.252.0303**
- In the United Kingdom, call Oracle Direct Response at: **+44.990.332200**
- In other European countries, contact your local Oracle Support office
- In the Asia-Pacific region, contact your Oracle sales representative

Shipping Inquiries

For shipping enquiries:

- In the United States, call Client Relations at: **1.650.506.1500**
- In the United Kingdom, call Customer Relations at: **+44.990.622300**
- In other European countries, contact your local Oracle Support office
- In the Asia-Pacific region, contact your Oracle sales representative

1

Optimal Flexible Architecture on Oracle8

Optimal Flexible Architecture (OFA) is a configuration standard that maximizes the speed and reliability of databases and minimizes the need for maintenance. This chapter describes the OFA standard and how to implement if for Oracle8 on UNIX systems. This chapter contains the following sections:

- Optimal Flexible Architecture
- OFA Implementation for Oracle8 on UNIX Systems

Optimal Flexible Architecture

Oracle Corporation recommends the Optimal Flexible Architecture (OFA) standard for Oracle8. The OFA standard is a set of configuration guidelines for creating fast, reliable Oracle databases that require little maintenance.

OFA is designed to:

- Organize large amounts of complicated software and data on disk to avoid device bottlenecks and poor performance
- Facilitate routine administrative tasks such as software and data backup functions, which are often vulnerable to data corruption
- Alleviate maintenance work between multiple Oracle databases
- Manage and administer database growth
- Eliminate fragmentation of free space in the data dictionary, isolate other fragmentation
- Minimize resource contention

Characteristics of an OFA-Compliant Database

The following sections describe the benefits of an OFA-compliant database.

File System Organization

The file system is organized to allow easy administration and accommodate scalability for:

- Adding data into existing databases
- Adding users
- Creating databases
- Adding hardware

Distributed I/O Loads

I/O loads are distributed across multiple disk drives to prevent performance bottlenecks.

Hardware Support

Hardware costs are minimized only when they do not conflict with performance considerations.

Safeguards Against Drive Failures

By spreading applications across more than one drive, drive failures impact as few applications as possible.

Distribution of Home Directories

The following items can be distributed across more than one disk drive:

- Home directories
- The contents of an individual home directory

Integrity of Login Home Directories

It is possible to add, move, or delete login home directories without having to revise programs that refer to them.

Independence of UNIX Directory Subtrees

Categories of files are separated into independent UNIX directory subtrees so that files in one category are minimally affected by operations on files in other categories.

Supports Concurrent Execution of Application Software

It is possible to execute multiple versions of applications software simultaneously. This allows the user to test and use a new release of an application before abandoning the previous version. Transferring to a new version after an upgrade is easy for the administrator and transparent for the user.

Distinguishes Administrative Information for Each Database

OFA-compliant databases have the ability to separate database administration for different databases. This facilitates organizing and storing administrative data.

Uses Consistent Database File Naming

Database files are named so that you can:

- Distinguish database files from all other files
- Distinguish files of one database from files of another database
- Identify control files, redo log files, and database files
- Recognize the association between database files and tablespaces

Separation of Tablespace Contents

Tablespace contents are separated to:

- Minimize tablespace free space fragmentation
- Minimize I/O request contention
- Maximize administrative flexibility

Tuning of I/O Loads Across All Drives

I/O loads are tuned across all drives, including drives storing Oracle data in raw devices.

OFA Implementation for Oracle8 on UNIX Systems

A careful naming strategy for database files helps avoid data-administration problems. The OFA rules provided here correspond to the original OFA recommendations published in *The OFA Standard: Oracle8 for Open Systems.*

Naming Mount Points

The following sections describe how to name mount points.

Mount Point Syntax

Name all mount points using the syntax */pm,* where *p* is a string constant and *m* is a unique fixed-length key that distinguishes each mount point. For example: `/u01` and `/u02`, or `/disk01` and `/disk02`.

Naming Mount Points for Very Large Databases (VLDBs)

If each disk drive contains database files from one application and there are enough drives for each database to ensure no I/O bottleneck, use the syntax */q/dm* to name mount points, where:

q is a string denoting that Oracle data is stored here

d is the value of the DB_NAME initialization parameter (synonymous with the instance *sid* for single-instance databases)

m is a unique, fixed-length key that distinguishes one mount point from another

For example, mount points named `/u01/oradata/test01` and `/u01/oradata/test02` allocate two drives for the Oracle test database.

Naming Directories

Name login home directories using the syntax */pm/h/u*, where:

pm is a mount point name

h is a standard directory name

u is the name of the owner of the directory

For example, `/u01/app/oracle` is the home directory of the `oracle` software owner and `/u01/app/applmgr` is the home directory of the Oracle applications software owner.

Placing home directories at the same level in the UNIX file system allows home directories to exist on different mount points. Refer to the collection of login home directories with a single pattern, such as `/*/app/*`.

Referring to Path Names

Refer to explicit path names only in files designed specifically to store them, such as `/etc/passwd` and the Oracle `/etc/oratab` file. Refer to group memberships only in the `/etc/group` file.

Software Directories

To fulfill the OFA requirement that multiple versions of application software execute simultaneously, store each version of Oracle8 software in a directory matching the pattern *h/product/v*, where:

h is a standard directory name

v is the version of the software

For example, `/u01/app/oracle/product/8.0.6` indicates the start of the directory structure where Oracle8 files are located. Set the ORACLE_HOME environment variable to this directory.

Naming Files

The following sections describe the conventions for naming files.

Administration Files

To facilitate the organization of administrative data, store database administration files in the sub-directories of *h*/admin/*d*/*a*, where *h* is the Oracle software owner's home directory, and *d* is the database DB_NAME initialization parameter, and *a* is the sub-directory for database administration files. The following table describes these sub-directories:

Subdirectory	Description
adhoc	Ad hoc SQL scripts for a given database
arch	Archived redo log files
adump	Audit files (Set AUDIT_FILE_DEST in config*db_name*.ora to point here. This subdirectory should be cleaned out periodically).
bdump	Background process trace files
cdump	Core dump files
create	Programs used to create the database
exp	Database export files
logbook	Files recording the status and history of the database
pfile	Instance parameter files
udump	User SQL trace files

For example, the subdirectory adhoc could have the following path name:

```
/u01/app/oracle/admin/sab/adhoc/
```

Database Files

Use the following naming convention for database files:

File Type	Naming Convention
Control files	*/pm/q/d/*`control.ctl`
Redo log files	*/pm/q/d/*`redo`*n*`.log`
Data files	*/pm/q/d/*`tn.dbf`

where:

pm is a mount point name described earlier in this chapter

q is a string distinguishing Oracle data from all other files (usually named ORACLE or oradata)

d is the value of the DB_NAME initialization parameter of the database

t is an Oracle tablespace name

n is a two-digit string

Note: Store only redo log or data files associated with database *d* in the directory */pm/q/d*.

This convention makes it easier to identify the database that the file belongs to. For example, a data file with the name, `/u03/oradata/sab/system01.dbf` belongs to the `sab` database.

Separate Segments with Different Requirements

Separate groups of segments with different life spans, I/O request demands, and backup frequencies across different tablespaces.

For each Oracle database, create the special tablespaces described in Table 1–1. These tablespaces are in addition to those needed for application segments.

Table 1–1 Special Tablespace

Tablespace	Segments
SYSTEM	Data dictionary segments
TEMP	Temporary segments
RBS	Rollback segments
TOOLS	General-purpose tools
USERS	Miscellaneous user segments

This method is effective because dictionary segments are never dropped, and no other segments that can be dropped are allowed in the SYSTEM tablespace. This ensures that you do not have to rebuild the SYSTEM tablespace because of the fragmentation of free space in the tablespace.

Because rollback segments are not stored in tablespaces holding applications data, the administrator is not blocked from taking an application's tablespace offline for maintenance. The segments are partitioned physically by type, and the administrator can record and predict data growth rates without complicated tools.

Naming Tablespaces

Name tablespaces descriptively using a maximum of eight characters.

Although Oracle8 tablespace names can be thirty characters long, portable UNIX file names are restricted to fourteen characters. The recommended standard for a data file basename is *tn*`.dbf`, where *t* is a descriptive tablespace name and *n* is a two-digit string. Because the extension plus the two-digit string occupy a total of six characters, only eight characters remain for the tablespace name.

Use descriptive names to allow the administrator to associate the name of a data file with the tablespace that uses it. For example, the names GLD and GLX might be used for the tablespaces storing General Ledger data and indices, respectively.

Note: Do not embed reminders of the word "tablespace" in your tablespace names. Tablespaces are distinguishable by context, and names do not need to convey information about type.

Using OFA Conventions for Oracle Files

Table 1–2 shows the syntax used for identifying classes of files.

Table 1–2 Directory Structure Syntax for Identifying Classes of Files

Directory Structure	File Classes
`/u[0-9][0-9]`	User data directories
`/*/home/*`	User home directories
`/*/app/*`	User application software directories
`/*/app/applmgr`	Oracle applications software subtrees
`/*/app/oracle/product`	Oracle Server software subtrees
`/*/app/oracle/product/8.0.6`	Oracle Server 8.0.6 distribution files
`/*/app/oracle/admin/sab`	`sab` database administrative subtrees
`/*/app/oracle/admin/sab/arch/*`	`sab` database archived log files
`/*/oradata`	Oracle data directories
`/*/oradata/sab/*`	`sab` database files
`/*/oradata/sab/*.log`	`sab` database redo log files

OFA File Mapping

Table 1–3 shows a hierarchical file mapping of a sample OFA-compliant database, including each file's mount point, application, database, and tablespace. The file names indicate the file type (control, log, or data).

Table 1–3 File Mapping for OFA Installation

File or Directory	Description
`/`	Root mount point
`/u01`	'User data' mount point #1
`/u01/app`	Subtree for application software
`/u01/app/oracle`	Home for `oracle` software owner
`/u01/app/oracle/admin`	Subtree for database admin files
`/u01/app/oracle/admin/TAR`	Subtree for support logs
`/u01/app/oracle/admin/TAR/`*`db_name1`*	Administrative subtree for *db_name1* database
`/u01/app/oracle/admin/TAR/`*`db_name2`*	Administrative subtree for *db_name2* database
`/u01/app/oracle/doc`	Online documentation
`/u01/app/oracle/local`	Subtree for local Oracle software
`/u01/app/oracle/local/aps6`	An Oracle6 administrative package
`/u01/app/oracle/local/aps7`	An Oracle7 administrative package
`/u01/app/oracle/product`	Distribution files
`/u01/app/oracle/product/7.3.2`	ORACLE_HOME for 7.3.2 instances
`/u01/app/oracle/product/7.3.3`	ORACLE_HOME for 7.3.3 instances
`/u01/app/oracle/product/8.0.5`	ORACLE_HOME for 8.0.5 instances
`/u01/app/oracle/product/8.0.6`	ORACLE_HOME for 8.0.6 instances
`/u01/home`	Subtree for login home directories
`/u01/home/ltb`	Home for a user
`/u01/home/sbm`	Home for a user
`/u01/oradata`	Subtree for Oracle data
`/u01/oradata/`*`db_name1`*	Subtree for *db_name1* database files
`/u01/oradata/`*`db_name2`*	Subtree for *db_name2* database files
`/u02`	'User data' mount point #2
`/u02/home`	Subtree for login home directories

Table 1–3 File Mapping for OFA Installation (Cont.)

File or Directory	Description
`/u02/home/cvm`	Home for a user
`/u02/home/vrm`	Home for a user
`/u02/oradata`	Subtree for Oracle data
`/u02/oradata/`*`db_name1`*	Subtree for *db_name1* database files
`/u02/oradata/`*`db_name2`*	Subtree for *db_name2* database files
`/u03`	'User data' mount point #3
`/u03/home`	Subtree for login home directories
`/u03/oradata`	Subtree for Oracle data
`/u03/oradata/`*`db_name1`*	Subtree for *db_name1* database files
`/u03/oradata/`*`db_name2`*	Subtree for *db_name2* database files

Raw Device Sizes

Choose a small set of standard sizes for all raw devices that might be used to store Oracle database files.

In general, it is best to standardize on a single size. If you specify a single size, raw files can be safely moved from one partition to another. Try to keep the standard size small enough to let you create several partitions but large enough for the partitions to be useful.

You could, for example, divide a 2 GB drive into 10 partitions of 200 MB each—a good balance between size and number. If you are using raw devices for a tablespace, you should stripe the raw devices across several drives. If possible, use a logical volume manager to configure the striping.

Directory Structure

The following sections describe the OFA directory structure for Oracle8 on SGI IRIX.

ORACLE_BASE Directory

ORACLE_BASE is the root of the Oracle directory structure. Table 1–4 describes ORACLE_BASE directory structure and content. When you install an OFA-compliant database using the Oracle Installer, ORACLE_BASE is set to the */pm*/app/oracle directory.

Table 1–4 ORACLE_BASE Directory Structure and Content

Subdirectory	Contents
admin	Administrative files
doc	Online documentation
local	Subtree for local Oracle software
product	Oracle software

ORACLE_HOME Directory

Table 1–5 describes ORACLE_HOME subdirectories and contents. When you install Oracle8 using the OFA standard, the ORACLE_HOME directory is */mount_point*/app/oracle/product/*release_number*.

On UNIX systems, the ORACLE_HOME directory contains the following subdirectories and a subdirectory for each installed Oracle product.

Table 1–5 ORACLE_HOME Directory Structure and Content

Subdirectory	Contents
bin	Binaries for all products
ctx	ConText cartridge
dbs	init*sid*.ora, lk*sid*
jdbc	JDBC drivers
lib	Oracle product libraries
md	Spatial cartridge
mlx	Xerox Stemmer (for ConText cartridge)
network	Net8

Table 1–5 ORACLE_HOME Directory Structure and Content (Cont.)

Subdirectory	Contents
nlsrtl	NLS runtime loadable data
ocommon	Common files for all products
oracore	Core libraries
orainst	Master installation files and programs
ord	Data cartridges
otrace	Oracle TRACE
plsql	PL/SQL
precomp	Precompilers
rdbms	Server files and libraries required for the database
slax	SLAX parser
sqlplus	SQL*Plus
svrmgr	Server Manager

Oracle Product Subdirectories

Depending on the Oracle products available on your platform and on which of these products you purchase and install, the $ORACLE_HOME directory might contain the subdirectories described in Table 1–6.

Table 1–6 Oracle Product Subdirectories

Subdirectory	Contents
network	Oracle Net8
ocommon	Libraries and SQL messages. All products depend on this directory, which is installed automatically
plsql	PL/SQL version 2, procedural option
sqlplus	SQL*Plus
svrmgr	Server Manager

Contents of Product Subdirectories

Each product subdirectory contains the subdirectories described in Table 1–7.

Table 1–7 Contents of Product Subdirectories

Subdirectory	Files
`admin`	Administrative SQL and shell scripts (for example, `catalog.sql`, `catexp.sql`, and `demo.sql`)
`admin/*`	Special directories for other products
`admin/resource`	Resource files
`admin/terminal`	Runtime terminal files
`demo`	Demonstration scripts and data files
`doc`	README files (for example, `readmeunix.doc`)
`install`	Product installation scripts
`lib`	Product libraries and distributed makefiles
`log`	Trace files and log files (for example, `orasrv.log` and `*.trc` files)
`mesg`	U.S. message files, and Multilingual Option (formerly National Language Support) message text and binary files (for example, `oraus.msg` and `oraus.msb`)

Examples of Product Subdirectories

Table 1–8 contains examples of product subdirectories and their contents.

Table 1–8 Examples of Product Subdirectories

Product Subdirectory	Contents
`rdbms`	`install`, `lib`, `admin`, `doc`, `mesg`, `log`
`sqlplus`	`install`, `demo`, `lib`, `admin`, `doc`, `mesg`

File Naming Conventions in the admin Directory

The `rdbms/admin` directory contains the SQL scripts shown in Table 1–9.

Table 1–9 admin Directory, File Naming Conventions

Script	Description

`cat*.sql`	Creates catalog and data dictionary tables and views. The Installer runs the following files automatically:
	`catalog.sql`
	`catproc.sql`

`dbms*.sql`	Additional database packages
`utl*.sql`	Creates tables and views for database utilities

File Name Extensions

Table 1-10 describes file name extensions.

Table 1-10 File Name Extensions

Extension	Description
`.aud`	Oracle audit file
`.bdf`	X11 font description file
`.bmp`	X11 bitmap file
`.c`	C source file
`.ctl`	SQL*Loader control file; Oracle Server control file
`.dat`	SQL*Loader datafile
`.dbf`	Oracle Server tablespace file
`.dei`	ORCA de-installation script
`.dmp`	Export file
`.doc`	ASCII text file
`.env`	Shell script file for setting environment
`.f`	Pro*FORTRAN source file
`.h`	C header file; also, `sr.h` is a SQL*Report Writer help file
`.ins`	ORCA installation script
.n (where *n* is any number)	UNIX manual page
`.lis`	Output of SQL*Plus scripts

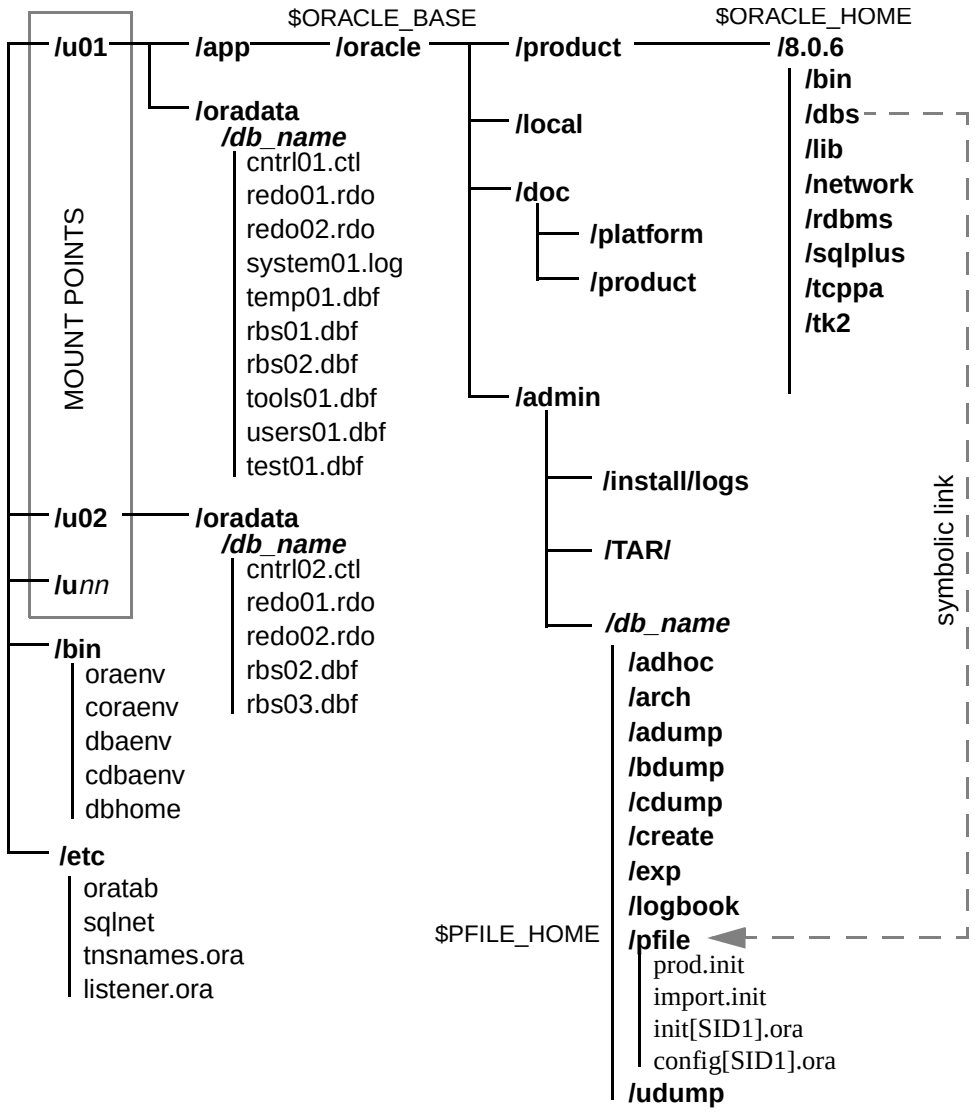
Table 1-10 File Name Extensions (Cont.)

Extension	Description
.log	Installation log files; Oracle Server redo log files
.map	Installer product component files
.mk	make files
.msb	NLS message file (binary)
.msg	NLS message file (text)
.o	Object module
.ora	Oracle configuration files
.orc	Installation prototype files
.pc	Pro*C source file
.pco	Pro*COBOL source file
.ppd	Printer driver file
.pfo	Pro*FORTRAN source file
.prd	Product registration template file (for orainst)
.res	Toolkit II resource file
.sh	Bourne shell script file
.sql	SQL* script files
.sys	Bourne shell script file
.tab	SQL* script file
.tut	Bourne shell script file
.us	orainst message file
.utd	Uniform Terminal Definitions file
.vrf	Installer Dependencies Verification script file

Default OFA Database

Figure 1–1 shows an OFA-compliant installation of Oracle8 created using the Oracle Installer.

Figure 1–1 Default Oracle Installation



2

Administering Oracle8 on SGI IRIX

The chapter describes how to administer Oracle8 on SGI IRIX systems. It contains the following sections:

- Customizing the initsid.ora File
- Setting the Environment
- Environment Variables for Oracle8
- Estimating Oracle8 Memory Usage
- Calculating Cluster Size and Index Size
- Initialization Parameters
- Tuning Oracle8 on IRIX Systems
- Controlling the System Global Area
- Managing Special Accounts and Groups
- Managing Security
- Administering Login Home Directories
- Building and Running Demonstrations
- Database Limits

Customizing the init*sid*.ora File

This section describes the default `init`*sid*`.ora` file provided with Oracle8. The Oracle Installer installs it in the `$ORACLE_BASE/admin/`*db_name*`/pfile` directory. You can modify it to customize your Oracle8 installation.

Some `init`*sid*`.ora` parameter settings are generic to any size installation. For those parameter settings requiring different values for different size installations, three scenarios are provided: small, medium, and large. In the sample `init`*sid*`.ora` file, parameters dependent on installation size are shown for each setting. You can comment out settings that do not apply to your installation by inserting a number sign (#) at the beginning of a line.

Table 2–1 suggests the approximate SGA sizes corresponding to the three scenarios provided for in the `init`*sid*`.ora` file.

Table 2–1 SGA Sizes for Sample initsid.ora File

Database Size	2 KB Block Size	4 KB Block Size
Small	4500 KB	5500 KB
Medium	6800 KB	8800 KB
Large	17000 KB	21000 KB

Sample init*sid*.ora File

This file is provided by Oracle Corporation to assist in customizing the RDBMS installation. Any parameter that needs to be tuned according to installation size will have three settings, each one commented according to installation size.

Example 2–1 Sample initsid.ora File

```
# replace DEFAULT with your database name
db_name=DEFAULT

db_files = 80                                                    # SMALL
# db_files = 400                                                 # MEDIUM
# db_files = 1000                                                # LARGE

db_file_multiblock_read_count = 8                                # SMALL
# db_file_multiblock_read_count = 16                             # MEDIUM
# db_file_multiblock_read_count = 32                             # LARGE

db_block_buffers = 100                                           # SMALL
```

```
# db_block_buffers = 550                                        # MEDIUM
# db_block_buffers = 3200                                       # LARGE

shared_pool_size = 3500000                                      # SMALL
# shared_pool_size = 5000000                                    # MEDIUM
# shared_pool_size = 9000000                                    # LARGE

log_checkpoint_interval = 10000

processes = 50                                                  # SMALL
# processes = 100                                               # MEDIUM
# processes = 200                                               # LARGE

parallel_max_servers = 5                                        # SMALL
# parallel_max_servers = 4 x (number of CPUs)                   # MEDIUM
# parallel_max_servers = 4 x (number of CPUs)                   # LARGE

log_buffer = 8192                                               # SMALL
# log_buffer = 32768                                            # MEDIUM
# log_buffer = 163840                                           # LARGE

sequence_cache_entries = 10                                     # SMALL
# sequence_cache_entries = 30                                   # MEDIUM
# sequence_cache_entries = 100                                  # LARGE

sequence_cache_hash_buckets = 10                                # SMALL
# sequence_cache_hash_buckets = 23                              # MEDIUM
# sequence_cache_hash_buckets = 89                              # LARGE

# audit_trail = true            # if you want auditing
# timed_statistics = true       # if you want timed statistics
max_dump_file_size = 10240      # limit trace file size to 5 Meg each

# Uncommenting the line below will cause automatic archiving if archiving has
# been enabled using ALTER DATABASE ARCHIVELOG.
# log_archive_start = true
# log_archive_dest = disk$rdbms:[oracle.archive]
# log_archive_format = "T%TS%S.ARC"

# If using private rollback segments, place lines of the following
# form in each of your instance-specific init.ora files:
# rollback_segments = (name1, name2)

# If using public rollback segments, define how many
# rollback segments each instance will pick up, using the formula
```

```
#   # of rollback segments = transactions / transactions_per_rollback_segment
# In this example each instance will grab 40/10 = 4:
# transactions = 40
# transactions_per_rollback_segment = 10

# Global Naming -- enforce that a dblink has same name as the db it connects to
global_names = TRUE

# Edit and uncomment the following line to provide the suffix that will be
# appended to the db_name parameter (separated with a dot) and stored as the
# global database name when a database is created.  If your site uses
# Internet Domain names for e-mail, then the part of your e-mail address after
# the '@' is a good candidate for this parameter value.

# db_domain = us.acme.com # global database name is db_name.db_domain

#_db_block_cache_protect = true                      # memory protect buffers
#event = "10210 trace name context forever, level 2" # data block checking
#event = "10211 trace name context forever, level 2" # index block checking
#event = "10235 trace name context forever, level 1" # memory heap checking
#event = "10049 trace name context forever, level 2" # memory protect cursors

# define parallel server (multi-instance) parameters
#ifile = ora_system:initps.ora

# define two control files by default
control_files = (ora_control1, ora_control2)

# Uncomment the following line if you wish to enable the Oracle Trace product
# to trace server activity.  This enables scheduling of server collections
# from the Oracle Enterprise Manager Console.
# Also, if the oracle_trace_collection_name parameter is non-null,
# every session will write to the named collection, as well as enabling you
# to schedule future collections from the console.

# oracle_trace_enable = TRUE
```

Setting the Environment

This section describes how to establish a common environment for your Oracle8 system.

Displaying and Setting Environment Variables

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

```
$ echo $ORACLE_SID
```

Setting and Exporting the Value of a Variable in a Current Session

The following examples show how to set and export variable values in the current session.

For the Bourne or Korn shell, enter:

```
$ ORACLE_SID=test
$ export ORACLE_SID
```

For the C shell, enter:

```
% setenv ORACLE_SID test
```

In both examples, *`test`* is the value of the variable ORACLE_SID.

Setting a Common Environment

Oracle8 enables a DBA to set a common environment for all users. A common environment makes it easier for system administrators and database administrators to make changes to the physical Oracle8 system.

The oraenv Command File

The `oraenv` (`coraenv` for the C shell) command file is created during installation. It contains values for Oracle environment variables and provides:

- A central means of updating all user accounts with database changes
- A mechanism for switching between Oracle databases

For example, you might find yourself frequently adding and removing databases from your development system or your users might be switching between several different Oracle databases installed on the same system. With `oraenv` (or `coraenv`), each user shell startup file calls the `oraenv` (or `coraenv`) command file.

Local bin Directory

Place `oraenv` (or `coraenv`) and `dbhome` scripts in a local `bin` directory, separate from the Oracle software home directory, to ensures that these files are accessible to all users. This also ensures that the `oraenv` (`coraenv`) script continues to work even if you change the path to point to a different `ORACLE_HOME` directory. The local `bin` directory is specified by the `root.sh` script, which you run after the installation. The default location for the local `bin` directory on SGI IRIX is `/usr/local/bin`.

Switching Between Databases

To switch from one database or database instance to another, call the `oraenv` routine. Reply to the prompt with the value of the $ORACLE_SID environment variable of the database to which you are switching. Always provide the full path of the `oraenv` command file. For example:

```
$ . /usr/local/bin/oraenv
ORACLE_SID= [default]? sid
```

Database Examples

In the following examples, it is assumed your local bin directory is called `/usr/lbin` and your production database is called PROD. If you prefer not to be prompted for the $ORACLE_SID environment variable at startup, set the ORAENV_ASK environment variable to NO.

In the following examples, ORAENV_ASK is reset to the default, YES, after `oraenv` is executed. This ensures that the system prompts for a different $ORACLE_SID the next time it executes the `oraenv` file.

Single Instance

For the Bourne or Korn shell, add or replace the following line in the `.profile` file:

```
. local_bin_directory/oraenv
```

with the following lines:

```
PATH=${PATH}:/usr/lbin
ORACLE_SID=PROD
export PATH ORACLE_SID
ORAENV_ASK=NO
. oraenv
ORAENV_ASK=
```

For the C shell, add or replace the following line in the `.cshrc` file:

```
source local_bin_directory/coraenv
```

with the following lines:

```
setenv PATH ${PATH}:/usr/lbin
setenv ORACLE_SID PROD
set ORAENV_ASK = NO
source /usr/lbin/coraenv
unset ORAENV_ASK
```

Multiple Instances

For multiple database instances, define the *sid* at startup.

For the Bourne or Korn shell:

```
PATH=${PATH}:/usr/lbin
ORACLE_SID=PROD
export PATH ORACLE_SID
SIDLIST= 'nawk -F: '/^[^#]/ {printf "%s ", $1}' /etc/oratab'
echo "SIDS on this machine are $SIDLIST"
ORAENV_ASK=
oraenv
```

For the C shell:

```
setenv PATH ${PATH}:/usr/lbin
setenv ORACLE_SID PROD
set sidlist = 'nawk -F: '/^[^#]/ {printf "%s ", $1}' /etc/oratab'
echo "SIDS on this machine are $sidlist"
unset ORAENV_ASK
source /usr/lbin/coraenv
```

Environment Variables for Oracle8

This section describes the most commonly used Oracle8 and UNIX environment variables.

> **See Also:** Refer to the *Oracle8 Installation Guide (32 Bit) for SGI IRIX* for more information on Oracle8 environment variables.

Oracle Environment Variables on UNIX

You must define some of these variables before you install Oracle8. These are listed in the *Oracle8 Installation Guide (32 Bit) for SGI IRIX*.

Table 2–2 provides the syntax and examples for Oracle8 variables.

Table 2–2 Oracle8 Environment Variables on UNIX

Variable	Detail	Definition
EPC_DISABLED	Function	Disables oracle trace.
	Syntax	`true` or `false`
	Example	`true`
NLS_LANG	Function	Specifies the language and character set used for output. See the *Oracle8 Installation Guide (32 Bit) for SGI IRIX* for a range of values.
	Syntax	*`language_territory.characterset`*
	Example	`french_france.we8dec`
ORA_NLS33	Function	Points to the directory where languages and character sets are stored.
	Default	`$ORACLE_HOME/ocommon/nls/admin/data`
ORACLE_BASE	Function	Specifies the base of the Oracle directory structure for OFA-compliant databases.
	Syntax	*`directory_path`*
	Example	*/`mount_point`/`app/oracle`*
ORACLE_HELP	Function	Specifies the directory containing help files.
	Syntax	*`directory_path`*
	Example	`$ORACLE_HOME/help/admin/resource`
ORACLE_HOME	Function	Specifies the directory containing the Oracle software distribution.
	Syntax	*`directory_path`*
	Example	*/`mount_point`*`/app/oracle/product/`*`release_number`*
ORACLE_PATH	Function	Specifies the search path name for files used by Oracle applications, such as SQL*Plus. If the full path name to the file is not specified or is not in the current directory, the application uses ORACLE_PATH to locate the file.
	Syntax	colon-separated list of directories *directory:directory:directory*
	Example	`/u01/oracle/adhoc/sqlplus:.`
		Note: The period adds the current working directory to the search path.

Table 2–2 Oracle8 Environment Variables on UNIX (Cont.)

Variable	Detail	Definition
ORACLE_SID	Function	Specifies the Oracle System Identifier.
	Syntax	The string of numbers and characters must begin with a letter. A maximum of four characters is recommended. For more information, see the *Oracle8 Installation Guide (32 Bit) for SGI IRIX*.
	Example	SAL1
ORACLE_TERM	Function	Specifies the terminal type identifier. Used by the Installer and Oracle products to determine the correct Toolkit II (`.res`) resource file. If not set, the value of the operating-system variable TERM is used.
	Syntax	String of characters.
	Range of Values	The value of this variable must be set such that the pattern `tk2c${ORACLE_TERM}.res` corresponds to valid resource files in the Toolkit II resource directory or directories. See the *Oracle8 Installation Guide (32 Bit) for SGI IRIX* for a list of valid values.
	Example	vt100
ORACLE_TERMINAL	Function	Specifies an additional directory to search for Toolkit II (`.res`) resource files.
	Syntax	*`directory_path`*
	Example	`$ORACLE_HOME/guicommon/tk21/admin/terminal`
ORACLE_TRACE	Function	Turns on tracing of Bourne shell scripts during install. If set to T, many Oracle shell scripts run with `set`-x flag on.
	Range of Values	T or anything else.
ORAENV_ASK	Function	Controls whether the `oraenv` or `coraenv` file prompts for values for ORACLE_SID or ORACLE_HOME. The `oraenv` or `coraenv` file prompts for these values unless this variable is set to `NO`.
	Syntax	String of characters.
	Range of Values	NO or anything else.
TNS_ADMIN	Function	Sets the directory containing the Oracle Net8 configuration files.
	Syntax	*`directory_path`*
	Range of Values	Any directory; for more information, see the *Oracle8 Installation Guide (32 Bit) for SGI IRIX*.
	Example	`$ORACLE_HOME/network/admin`

Table 2–2 Oracle8 Environment Variables on UNIX (Cont.)

Variable	Detail	Definition
TWO_TASK	Function	Sets the default Oracle Net8 connect string descriptor alias defined in the `tnsnames.ora` file.
	Syntax	String of characters.
	Range of Values	Any valid Oracle Net8 alias defined in the `tnsnames.ora` file.
	Example	PRODDB_TCP

Note: If ORACLE_TERM is not set correctly, your screen display and function keys might not function properly.

Note: Environment variables should not be defined with names that are identical to names of Oracle Server processes, for example: `arch`, `pmon`, and `dbwr`.

In Oracle8 files and programs, a question mark (?) represents the value of the ORACLE_HOME environment variable. For example, Oracle8 expands the question mark in the following SQL statement to the full path name of ORACLE_HOME:

```
ALTER TABLESPACE TEMP ADD DATAFILE '?/dbs/dbs2.ora' SIZE 2M
```

The at sign (@) represents the value of the `$ORACLE_SID` environment variable. For example, to indicate that a file belongs to an instance, enter:

```
ALTER TABLESPACE tablespace_name ADD DATAFILE 'dbsfile@.ora'
```

UNIX Environment Variables Used with Oracle8

Table 2–3 provides the syntax and examples for UNIX environment variables used with Oracle8.

Table 2–3 UNIX Environment Variables Used with Oracle8

Variable	Detail	Definition
DISPLAY	Function	Used by X-based tools. Specifies the display device used for input and output. See vendor's X Windows documentation for details.
	Syntax	*hostname*:*display* where *hostname* is the network identifier for the display device *display* is a number which is almost always 0.
	Example	135.287.222.12:0 bambi:0
HOME	Function	The user's home directory.
	Syntax	*directory_path*
	Example	`$ HOME`
LANG or LANGUAGE	Function	Specifies the language and character set used by the operating system for messages and other output. See the operating system documentation, and your *Oracle8 Installation Guide (32 Bit) for SGI IRIX*.
LDOPTS	Function	Specifies the default linker options on some platforms. See `man` pages on `ld` for details.
LPDEST	Function	Specifies the user's default printer.
	Syntax	*printer_name*
	Example	`docqms`
LD_LIBRARY_PATH	Function	Used on some platforms by the shared library loader (`rld`) at runtime to find shared object libraries. See `man` pages on `rld` for details.
	Syntax	colon-separated list of directories *directory*:*directory*:*directory*
	Example	`/usr/dt/lib:$ORACLE_HOME/lib`
PATH	Function	Used by the shell to locate executable programs; needs to include `$ORACLE_HOME/bin`.
	Syntax	colon-separated list of directories *directory*:*directory*:*directory*

Table 2–3 UNIX Environment Variables Used with Oracle8 (Cont.)

Variable	Detail	Definition
	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.
SHELL	Function	Specifies the command interpreter used during a host command.
	Syntax	shell path name
	Range of Values	`/bin/sh` or `/bin/csh` or `/bin/ksh` or any other command interpreter supplied with your system.
	Example	`/bin/sh`
TERM	Function	Used by Oracle Toolkit II character mode tools to determine terminal types; also used by other UNIX tools for the same purpose.
	Example	vt100
TMPDIR	Function	Specifies the default directory for temporary disk files; if set, tools that create a temporary files do so in this directory.
	Syntax	*`directory_path`*
	Example	`/u02/oracle/tmp`
XENVIRONMENT	Function	Specifies a file containing X Windows system resource definitions. See your X Windows documentation for more information.

Setting the System Time

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

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

Estimating Oracle8 Memory Usage

You need to know Oracle8's memory usage requirements before starting the Installer. Knowing these requirements helps you determine the number of users you can have on your system, and helps you determine your physical memory and swap space requirement. To calculate the memory requirements, use the following formula:

size of the oracle executable text
+ size of the SGA
+(number of background processes) * (size of tool executables private data section)
+size of oracle executables uninitialized data section

+ 8192 bytes for the stack
+ 2048 bytes for the processes user area

To determine the SGA size, see "Calculating the Size of the SGA" on page 2-24.

For each Oracle back-end connection, use the following formula to estimate virtual memory requirements:

size of oracle executable data section
+ size of oracle executables uninitialized data section

+ 8192 bytes for the stack
+ 2048 bytes for the processes user area

+ cursor area needed for the application

Use the `size` command to estimate an executable's text size, private data section size, and uninitialized data section size. Program text is counted only once, no matter how many times the program is run because all Oracle executable text is always shared.

To calculate actual Oracle physical memory usage while the database is running and users are connecting to it, use the `ps -elf` command. Look for all of the front end, server, and background Oracle process entries. For each entry, total the "SZ" columns.

> **See Also:** See your operating system `man` pages or documentation for a list of available switches for the `ps` command.

The `ps` command returns process size in pages; your system page size is architecture-dependent. Use the `pagesize` command to determine whether the size is 4096 or 8192 bytes. For each process, multiply the SZ value by the page size.

Finally, add the text size for the Oracle executable and every other Oracle tool executable running on the system to that subtotal. Remember to count executable sizes only once, regardless of how many times the executable was run.

Calculating Cluster Size and Index Size

Use the values in the tables in this section with the information in the *Oracle8 Administrator's Guide* to calculate the sizes of your database clusters and index.

Calculating Cluster Size

Use the size guidelines listed in Table 2–4 to calculate cluster size using the formula in Appendix A of the *Oracle8 Administrator's Guide.*

Table 2–4 Cluster Size Values

Type	Size
Fixed header size	68 bytes
Variable transaction header	24**INITRANS* value for the table
Row directory	4 bytes per row of a clustered table

Calculating Database Limits

Use the size guidelines listed in Table 2-5 to calculate database limits.

> **See Also:** For more information on calculating database limits, refer to the *Oracle8 Administrator's Guide.*

Table 2-5 Calculating Database Limits

Type	Size
Database file size	5 MB
Database files	255 or the value of the DB_FILES parameter in the `init.ora` file
Instances	255
Locks	100 (DML_LOCKS parameters value)
MAXEXTENTS	57 for a 1 KB block size 121 for a 2 KB block size 249 for a 4 KB block size 505 for an 8 KB block size
MAXEXTENTS	57 for a 1 KB block size 121 for a 2 KB block size 249 for a 4 KB block size 507 for an 8 KB block size
Redo log files	255

Data files are located in the `$ORACLE_HOME/dbs` directory by default.

Calculating Index Size

Use the size guidelines listed in Table 2–6 to calculate the size required by an index using the formula in Appendix A of the *Oracle8 Administrator's Guide.*

Table 2–6 Index Size Values

Type	Size
Fixed header size	113 bytes
Variable transaction header	24**INITRANS* value for the index
Entry header	5 bytes

Initialization Parameters

You can modify initialization parameters in the `init`*`sid`*`.ora` file for the Oracle8 instance.

Default Initialization Parameter Values

Table 2–7 lists default initialization parameter values on SGI IRIX systems. All Oracle8 instances assume these values unless you specify different values for them in the `init`*`sid`*`.ora` file. Oracle Corporation recommends that you include in the `init`*`sid`*`.ora` file only those parameters that differ from the default initialization parameter values.

To display the current values of these parameters on the system, use Server Manager to execute the SQL statement SHOW PARAMETERS.

> **See Also:** For more information on default initialization parameters, see the *Oracle8 Administrator's Guide.*

Table 2–7 Default Initialization Parameters

Parameter	Default Value
AUDIT_FILE_DEST	`$ORACLE_HOME/rdbms/audit`
BACKGROUND_DUMP_DEST	`$ORACLE_HOME/rdbms/log`
BITMAP_MERGE_AREA_SIZE	1048576
COMMIT_POINT_STRENGTH	1

Table 2–7 Default Initialization Parameters (Cont.)

Parameter	Default Value
CONTROL_FILES	`$ORACLE_HOME/dbs/ctrl@.dbf` (where @ represents ORACLE_SID)
CREATE_BITMAP_AREA_SIZE	8388608
DB_BLOCK_BUFFERS	100
DB_BLOCK_SIZE	2048
DB_FILES	30 (maximum of 1022)
DB_FILE_DIRECT_IO_COUNT	64 blocks (maximum of 1048576)
DB_FILE_MULTIBLOCK_READ_COUNT	8 (range of 1-128, but should not exceed one quarter of DB_BLOCK_BUFFERS)
DISTRIBUTED_TRANSACTIONS	16
HASH_AREA_SIZE	0
HASH_MULTIBLOCK_IO_COUNT	1
LOCK_SGA	FALSE
LOCK_SGA_AREAS	0
LOG_ARCHIVE_BUFFER_SIZE	64 blocks
LOG_ARCHIVE_BUFFERS	4 (maximum of 128)
LOG_ARCHIVE_DEST	`$ORACLE_HOME/dbs/arch/`
LOG_ARCHIVE_FORMAT	“%t_%s.dbf”
LOG_BUFFER	512 Kilobytes
LOG_CHECKPOINT_INTERVAL	1000
LOG_SMALL_ENTRY_MAX_SIZE	80
MTS_MAX_DISPATCHERS	5
MTS_MAX_SERVERS	20
MTS_SERVERS	0
MTS_LISTENER_ADDRESS	ADDRESS=*address* (See Chapter 6)
NLS_LANGUAGE	AMERICAN
NLS_TERRITORY	AMERICA
OBJECT_CACHE_MAX_SIZE_PERCENT	10
OBJECT_CACHE_OPTIMAL_SIZE	102400
OPEN_CURSORS	50

Table 2–7 Default Initialization Parameters (Cont.)

Parameter	Default Value
OS_AUTHENT_PREFIX	ops$
PROCESSES	25
SHARED_POOL_SIZE	3500 Kilobytes
SORT_AREA_SIZE	65536
SORT_READ_FAC	5
SORT_SPACEMAP_SIZE	512
TEMPORARY_TABLE_LOCKS	32
TRANSACTIONS_PER_ROLLBACK_SEGMENT	16
USER_DUMP_DEST	`$ORACLE_HOME/rdbms/log`
USE_DIRECT_IO	`FALSE`

Log Files

USE_DIRECT_IO

Default value: FALSE

Range of values: TRUE, FALSE

If the USE_DIRECT_IO parameter is set to TRUE, file read and writes are performed directly to and from Oracle server data buffers, bypassing the buffer cache maintained by SGI IRIX. Raw device input and output is unaffected. Using this option might improve or worsen performance depending on the application and system configuration.

SGA

USE_ISM

Default value: TRUE

Range of values: TRUE, FALSE

The USE_ISM parameter controls the use of shared page tables. On platforms that support it, setting USE_ISM to TRUE causes different processes to use the same set of page tables to reference the SGA.

This might result in a performance improvement, because context switches should become less expensive. However, individual processes can no longer write-protect individual regions of the SGA, so this should not be used for debugging.

LOCK_SGA

Default value: FALSE

Range of values: TRUE, FALSE

The LOCK_SGA parameter locks the entire SGA into physical memory. Platforms that do not support this parameter ignore it. For more information, see "Lock the SGA in Physical Memory" on page 3-5.

Post Wait Driver

Oracle processes usually use semaphores to coordinate access to shared resources. If a shared resource is locked, a process suspends and waits for the resource to become available.

One way to improve shared resource coordination is to use a post-wait driver instead of semaphores, if it is available on your system. A post-wait driver is a faster, less expensive synchronization mechanism than a semaphore. The Oracle Post-Wait driver implements an optimized mechanism of inter-process communication, without the overhead of signal handlers or semaphores. It improves performance for Oracle8.

USE_POST_WAIT_DRIVER

Default value: FALSE

Range of values: TRUE, FALSE

Set this parameter to TRUE to enable the post/wait driver, a lighter-weight alternative to semaphores. Oracle Corporation recommends using this option because it usually improves performance.

POST_WAIT_DEVICE

Default value: `/dev/postwait`

Range of values: any valid post/wait device name

This parameter stores the name of the post/wait device. There is rarely any need to change it from its default value.

IRIX-Specific Parameters

If you plan to use the post/wait driver on SGI IRIX, your SGI IRIX system administrator might need to change the following IRIX system parameters in the files that relate to the post/wait driver. The default values for these parameters are specified in the `/var/sysgen/master.d/postwait` file:

PW_MAXINST This parameter limits the number of post/wait driver instances and limits the number of Oracle Server instances using the driver. There is a one-to-one mapping between the Oracle Server instance and post/wait driver instance when using the driver.

Default Value: 4

PW_MAXENT This parameter limits the number of processes that can take part in a post/wait instance, therefore limiting the number of Oracle Server instances. Set this parameter to the largest value given to the PROCESS initialization parameter of any Oracle Server instance on the system.

Default Value: 512

PW_TIMER This parameter controls the timing cycle of the post/wait driver. Do not alter this parameter without explicit direction from SGI to do so.

Tuning Oracle8 on IRIX Systems

The following sections describe how to tune Oracle8 on SGI IRIX systems:

IRIX_CPU_AFFINITY

This initialization parameter enables you to specify processor affinity within an IRIX system. This parameter, which is available only on systems with the IRIX-specific enhancements installed, allows more control over system resources.

Default value: "all=any"

Syntax: IRIX_CPU_AFFINITY="NAME=MASK [NAME=MASK...]"

where:

MASK is a single CPU number or a range of CPU numbers or the special keyword 'ANY' which denotes the sysmp(2) option MP_RUNANYWHERE. Specify a range of CPU numbers in the form start_number:end_number where the start_number is less than the end_number.

NAME is one or more of the following values:

Value	Description
ALL	Apply the MASK to all Oracle8 processes not covered by a previously specified NAME=MASK combination.
OTHER	Apply MASK to all Oracle8 processes not covered by the sys NAME.
SYS	Apply the MASK to any Oracle8 system processes not covered by another specified NAME=MASK combination.
PMON	Apply the MASK to `pmon` Oracle8 system process.
DBW*N*	Apply the MASK to `dbw`*n* Oracle8 system process.
LGWR	Apply the MASK to `lgwr` Oracle8 system process.
SMON	Apply the MASK to `smon` Oracle8 system process.
CKPT	Apply the MASK to `ckpt` Oracle8 system process.
ARCH	Apply the MASK to `arch` Oracle8 system process.
RECO	Apply the MASK to `reco` Oracle8 system process.
LOCK	Apply the MASK to `lock` Oracle8 system process.
SNP*N*	Apply the MASK to `snp`*n* Oracle8 system process.
QMN*N*	Apply the MASK to `qmn`*n* Oracle8 system process.

Examples

The following specification of the IRIX_CPU_AFFINITY parameter assigns the PMON, SMON and CKPT processes to CPU 1, the DBWR process to CPU 2 and the LGWR process is to CPU 3:

```
IRIX_CPU_AFFINITY="pmon=1 dbwr=2 smon=1 lgwr=3 ckpt=1 other=4:7"
```

All other Oracle8 processes are assigned in round-robin fashion (based on Oracle process id) to CPU 4 through 7.

The following specification of the IRIX_CPU_AFFINITY parameter assigns all the Oracle8 system processes to CPUs 1 through 4 in round-robin fashion based on the Oracle process id and assigns all other Oracle8 processes to CPUs 5 through 19 in round-robin fashion based on the Oracle process id:

IRIX_CPU_AFFINITY="sys=0:4 other=5:19"

IRIX_SCHEDULER

This initialization parameter allows you to specify certain IRIX scheduler options.

> **Note:** To use IRIX_SCHEDULER, the `oracle` instance owner must have root privileges.

Default value: none

Syntax: IRIX_SCHEDULER="NAME=MASK [NAME=MASK...]"

where:

MASK is a numeric value valid for the particular NAME value.

NAME is one or more of the following values:

Value	Description
`ndpri`	Apply the MASK as the `schedctl(2)` NDPRI value.
`renice`	Apply the MASK as the `schedctl(2)` RENICE value.
`slice`	Apply the MASK as the `schedctl(2)` SLICE value.

The `$ORACLE_HOME/bin/oracle` executable might require you to set super user ID `root` privilege, depending on the scheduler options used.

Refer to the `schedctl(2)` man page for the details of the `schedctl(2)` command.

Examples

The following examples demonstrate using the IRIX_SCHEDULER parameter:

IRIX_SCHEDULER="slice=50"

IRIX_SCHEDULER="ndpri=39 slice=50"

Controlling the System Global Area

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

Size Limits of the SGA

Oracle8 uses shared memory segments for the SGA.

The maximum size of a single shared memory segment is specified by the SGI IRIX parameter SHMMAX. For example, if SHMMAX is 132 MB and the SGA is 528 MB, the SGA requires four segments. This setting is located in the `/var/sysgen/mtune/kernel` directory.

If the size of the SGA exceeds the maximum size of a shared memory segment (SHMMAX), Oracle8 attempts to attach more contiguous segments to fulfill the requested SGA size. SHMSEG is the maximum number of segments that can be attached by a process. To attach the segments at contiguous addresses, SHMMAX must be set to its maximum value on systems where its size is limited.

Note: Shared PTE can cause problems when SHMMAX is smaller than the database SGA size.

Set the following initialization parameters to control the size of the SGA:

- DB_BLOCK_BUFFERS
- DB_BLOCK_SIZE
- SORT_AREA_SIZE
- SHARED_POOL_SIZE

Be careful when setting values for these parameters. When values are set too high, too much of the computer's physical memory is devoted to shared memory, adversely affecting performance. As a guideline, the total of all instance's SGA sizes should be no more than one-third of the total physical RAM.

Calculating the Size of the SGA

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

- You can calculate the approximate size of an instance's SGA using the following formula:

  (DB_BLOCK_BUFFERS x DB_BLOCK_SIZE)
  + SORT_AREA_SIZE
  + SHARED_POOL_SIZE
  + LOG_BUFFER
  + JAVA_POOL_SIZE

- To display the size of the SGA for a running database, in bytes, use the SQL*Plus SHOW SGA command.
- You can also find the size of the SGA when you start the database system. The SGA size is displayed next to the heading Total System Global Area.

Relocating the SGA

The address at which the SGA is attached affects the amount of virtual address space available for such things as database buffers in the SGA and cursors in the user's application data area.

To determine the valid virtual address range for attaching shared memory segments, use the `tstshm` executable included in this release of Oracle8:

The address at which the SGA is attached affects the amount of virtual address space available for such things as database buffers in the SGA and cursors in the user's application data area.

1. Determine the valid virtual address range for attaching shared memory segments (in the resulting tstshm display, the lines "Lowest shared memory address" and "Highest shared memory address" indicate the valid range):

   ```
   $ tstshm
   ```

   Note: The system may experience problems when executing `tstshm` while using Intimate Shared Memory (ISM).

2. Check the "Segment boundaries" output of tstshm to determine the valid virtual address boundaries at which a shared memory segment can be attached.
3. Determine the size of your SGA. SGA size is displayed next to the heading Total System Global Area when your database system starts.
4. Change to the `$ORACLE_HOME/rdbms/lib` directory, and run `genksms` to generate the file `ksms.s`:

   ```
   $ cd $ORACLE_HOME/rdbms/lib
   $ $ORACLE_HOME/bin/genksms -b sgabeg > ksms.s
   ```

   where *sgabeg* is the starting address of the SGA (which defaults to 0x30000000 on SGI RIX), and should fall within the range determined in step 2.
5. Shut down the existing Oracle database.

6. Rebuild the `oracle` executable in the `$ORACLE_HOME/rdbms/lib` directory:

```
$ make -f ins_rdbms.mk ksms.o
$ make -f ins_rdbms.mk ioracle
```

Using `ioracle`:

- backs up the old executable (`oracle0`)
- assigns the correct privileges to the new `oracle` executable
- moves the new executable into the `$ORACLE_HOME/bin` directory

The result is a new Oracle kernel that loads the SGA at the address specified by *sgabeg*.

Managing Special Accounts and Groups

The DBA should be familiar with the special accounts required for administering the Oracle Server, and should make sure these accounts belong to the appropriate groups.

Special Accounts

Table 2–8 describes UNIX accounts. Table 2–9 describes Oracle server accounts. Table 2–10 describes special UNIX accounts.

Table 2–8 UNIX Accounts

Account	Description
oracle	The *`oracle`* software owner represents the account that owns the Oracle8 software. This maintenance account requires DBA privileges to run the CREATE, STARTUP, SHUTDOWN, and CONNECT as INTERNAL commands. The *`oracle`* software owner must never be the superuser.
`root`	The `root` user is a special UNIX account with maximum privileges (called superuser privileges). This account is used to configure the UNIX kernel, configure and install networking software, and create user accounts and groups.

Table 2–9 Oracle Server Accounts

Account	Description
SYS	This is a standard Oracle8 account with DBA privileges automatically created during installation. The SYS account owns all of the base tables for the data dictionary. This account is used by the DBA.
SYSTEM	This account is also a standard Oracle8 account, with DBA privileges automatically created during installation. Additional tables or views can be created by the SYSTEM user. DBAs can log in as SYSTEM to monitor or maintain databases.

Special Groups

Table 2–10 describes special groups.

Table 2–10 Special Groups

Groups	Description
`dba`	The *`oracle`* software owner is the only required member of the `dba` group. You can add other UNIX users, to the `dba` group. Members of this group have access to Server Manager specially privileged functions. If your account is not a member of the `dba` group, you must enter a password to connect as INTERNAL or gain access to the other administrative functions of Server Manager. The default group ID is `dba`.
`oper`	Members of this optional UNIX group have database OPERATOR privileges. OPERATOR privileges are a restricted set of DBA privileges.
`root`	Only the `root` user should be a member of the `root` group.

Managing Security

Oracle8 uses several features of the UNIX operating system to provide a secure environment for users. These features include file ownership, group accounts, and the ability of a program to change its user ID upon execution.

The two-task architecture of Oracle8 improves security by dividing work (and address space) between the user program and the `oracle` program. All database access is achieved through the shadow process and special authorizations on the `oracle` program.

Groups and Security

To ensure greater security on an Oracle8 database, create user groups at the operating system level. Groups are controlled by the UNIX file `/etc/group`. Oracle programs are divided into two sets for security purposes: those executable by all (*other*, in UNIX terms), and those executable by DBAs only. A recommended approach to security is:

- Before installing the Oracle Server, create a database administrators' group *(*`dba`*)* and assign the `root` and *`oracle`* software owner IDs to this group. Programs executable by `dba` only have permission `710`. Server Manager system-privileged commands are assigned automatically to the `dba` group upon installation.
- Add an `oracle` group of authorized users to allow a subset of UNIX users limited access to Oracle8. Give Oracle utilities the `oracle` group ID. Publicly executable programs, such as SQL*Plus, should be executable by this group. Set the permissions on the utilities to `710` to grant execute permissions to this group, but not *other*.
- Grant permission `711` to programs executable by *other*. Restrict this permission to programs that do not affect database security.

Although you can assign any name to the database administrators' group, `dba` is the default group name, and the convention used in this document. If you change this group name, the Oracle Installer relinks the kernel automatically during Installation. If you have multiple databases with the same ORACLE_HOME environment variable value (a configuration which Oracle Corporation *strongly* discourages), they should have the same database administrator group. These restrictions do not apply to the group name for ordinary users (known as the `oracle` group).

Note: Even though both the *`oracle`* software owner and `root` user should belong to the `dba` group, the *`oracle`* software owner should not be a member of the `root` group. The `root` user should be the only member of the `root` group.

Security for Oracle Server Utilities

Protect the Oracle8 executables from unauthorized use. The method you use depends on your environment and whether you use single-task utilities. The following are some suggestions for protecting Oracle8 executables:

- Keep all programs in the `$ORACLE_HOME/bin` directory and give ownership to the *`oracle`* software owner.
- Give all user utilities (`sqlplus, exp, imp`) a protection of `711` so that all users on the system can access the Oracle Server.
- Give all DBA utilities (such as Server Manager) a protection of `700` to restrict the use of these utilities to the DBA user name, usually the *`oracle`* software owner.

Security for Server Manager Commands

If you do not have SQL*Plus, you can use Server Manager to make SQL queries. However, be careful how you assign access to Server Manager. The following system-privileged statements should not be accessible to anyone but the `oracle` software owner and the `dba` group users, as they grant special operating system privileges:

- STARTUP
- SHUTDOWN
- CONNECT INTERNAL

CAUTION: System-privileged statements can damage your database if used incorrectly. Note that non-DBA group users can connect as INTERNAL if they have the necessary password.

Security for Database Files

The user ID used to install Oracle8 should own the database files. The default user ID is the *`oracle`* user. Set the authorizations on the database files to 0600 which permits read / write (rw) by owner only, with no write authorizations for group or other users.

The *`oracle`* software owner should own the directories containing the database files. For added security, revoke read permission from group and other users.

To access the protected database files, the `oracle` program must have its set user ID (`setuid`) bit on.

The Installer automatically sets the permissions of the `oracle` executable to:

```
-rwsr-s--x
```

The `s` in the user execute field means that the `oracle` program has an effective user ID of `oracle`, regardless of the actual user ID of the person invoking it.

If you need to set the permissions on the `oracle` executable manually, enter:

```
$ chmod 6751 $ORACLE_HOME/bin/oracle
```

Setting the User ID

The Installer automatically sets the user ID. The `s` in the user execute field means when you execute the `oracle` program, it has an effective user ID of `oracle`, regardless of the actual user ID of the person invoking it.

Network Security

This section describes network security issues on Oracle8.

Using Passwords on the Network

Remote users on the network can enter passwords in clear or encrypted text. When you use clear text, passwords can be picked up by unauthorized users, resulting in a breach of security. Oracle Net8 supports encrypted passwords.

DBA Privileges Over the Network

To control DBA privileges over the network choose one of the following options:

- Set remote DBA access to `denied` in the `/var/opt/oracle/listener.ora` file
- Set a special password in `orapwd` for DBA privileges

Automatic Logins (OPS$)

Oracle8 supports automatic logins (operating system authorized logins) over the network.

UNIX treats a dollar sign ($) as the beginning of an environment variable. Therefore, when you specify an operating system authorized (`ops$`) login on the command

line or in a script, first escape the $ with a backslash (\). For example, user ID SCOTT should specify ops\$SCOTT when logging in remotely.

Automatic logins are not allowed for the root user ID.

Note: Automatic logins by PC, Apple MacIntosh, and OS/2 users are not secure. Anyone can edit the Oracle configuration file and change their user ID. For security reasons, if users of these systems are logging in over the network, Oracle Corporation strongly recommends that you disable the ops$ logins in the listener.ora file.

Enabling Automatic Logins for Oracle Net8

Automatic and remote DBA logins are not controlled by Oracle Net8. They are controlled by Oracle8 and configured using parameters in the init*sid*.ora file. Although automatic logins are supported, they are disabled by default. To enable them, set the REMOTE_OS_AUTHENT initialization parameter to true, then start the database.

Because the *oracle* account controls these logins, you do not need use the root account to to run the Oracle Net8 listener.

> **See Also:** See Chapter 6 for more information on configuring Oracle Net8.

To perform an automatic login with Oracle Net8, create a user called daemon in your /etc/passwd file. The daemon user must not have an ops$ account in any of the local databases, nor be in any of the DBA groups. That is, there should be no ops$daemon account that would allow an outside user to intrude into your local database.

DBA Group ID Keywords

Table 2–11 describes the keywords used in the `/var/opt/oracle/listener.ora` file to enable and control remote logins.

Table 2–11 Keywords Used to Control Logins

Keyword	Description
DBA_GROUP	Use this keyword if the name is constant for all instances serviced by the listener.
DBA_GROUP_*sid*	Use this keyword for each ORACLE_SID environment variable if the listener services more than one `$ORACLE_HOME` directory, and the group IDs are different.
OPS_DOLLAR_LOGIN_ALLOWED OPS_DOLLAR_LOGIN_DENIED	Use these keywords to control remote login. OPS_DOLLAR_LOGIN_DENIED is the default.
REMOTE_DBA_OPS_ALLOWED REMOTE_DBA_OPS_DENIED	Use these keywords to control remote DBA access. REMOTE_DBA_OPS_DENIED is the default.

You can specify the name of the DBA group for the database that you are accessing. This is not necessary if you use the default name (`dba`).

To specify the DBA group name, set remote login and remote DBA access parameters to the individual ORACLE_SIDs of databases on the network, or specify all *sid*s at once. For example, either of the following statements are valid:

```
PARAMETER=ALL_SIDS
PARAMETER=sid1[, sidn...]
```

To see which privileges are assigned to the *sid*s, enter:

```
$ lsnrctl status
```

Checking Order

The system checks remote login parameters in the following order:

- Parameters that deny access
- Parameters that permit access
- The default value (denied)

These privileges are implemented by manipulating the user ID and group ID of the shadow process forked by the Oracle Net8 listener. For example:

- If the value of the OPS_DOLLAR_LOGIN_DENIED parameter is `true` for a particular instance, or if the user ID as reported by the client-side operating system has no account on the database host system, the user ID and group ID are found in the `/etc/passwd` file under the entry for `daemon`.
- If both the values of the OPS_DOLLAR_LOGIN_ALLOWED and the REMOTE_DBA_OPS_ALLOWED parameters are `true` for a particular system identifier**,** and if the user ID as reported by the client operating system does have an account on this system, the user ID and group ID value are found in `/etc/passwd directory` for this user ID.
- If the value of the OPS_DOLLAR_LOGIN_ALLOWED parameter is `true` for a particular system identifier, but the value of the REMOTE_DBA_OPS_ALLOWED parameter is `false`, then, if the user ID has DBA privileges, the process has the user ID and group ID of `daemon`. Otherwise, the process has the user ID and group ID of this user

> **Note:** The value of the REMOTE_DBA_OPS_ALLOWED parameter is `false` by default. Oracle Corporation recommends that you do not change this value. When this parameter is set to `false`, users with DBA privileges cannot make operating system authorized logins over the network. They can, however, proceed with ordinary (password-protected) network logins.

Security and Remote Passwords

You can access or administer a database from a remote system, without operating system accounts. User validation is accomplished by using an Oracle8 password file, created and managed by the `orapwd` utility. You can also use password file validation on systems that support operating system accounts.

Local password files are in the `$ORACLE_HOME/dbs` directory and contain the user name and password information for a single database. If there are multiple `$ORACLE_HOME` directories on a system, each has a separate password file. To allow the database to use the password file, set the `init`*`sid`*`.ora` parameter REMOTE_LOGIN_PASSWORDFILE to EXCLUSIVE.

Running orapwd

The `orapwd` utility exists in the `$ORACLE_HOME/bin` directory and is run by the `oracle` software owner. Invoke `orapwd` by entering:

```
$ orapwd file=$ORACLE_HOME/dbs/orapwsid password=password \
 entries=max_users
```

This syntax is described in the following table:

`filename`	is the name of the file where password information is written. The name of the file must be `orapw`*`sid`*, and you must supply the full path name. Its contents are encrypted and not user-readable. This parameter is mandatory.
`password`	is the initial password you selected for INTERNAL and SYS. You can change this password after you create the database using an ALTER USER statement. This parameter is mandatory.
`max_users`	is the maximum number of users allowed to connect to the database as SYSDBA or SYSOPER. This parameter is mandatory only if you want this password file to be EXCLUSIVE. Set *`max_users`* to a higher number than you expect to require because if you need to exceed this value, you must create a new password file.

Note: You must create a new password file if you ever need to increase the maximum number of users. Therefore, set *max_users* to a higher number than you expect to require.

orapwd Example

```
$ orapwd file=/u01/app/oracle/product/8.0.6/dbs/orapwV806 \
 password=manager entries=30
```

> **See Also:** For more information on security and remote passwords, refer to the *Oracle8 Administrator's Guide*.

Shared Password File for Multiple Databases

The default password file `/dbs/orapwd` should be used when the initialization parameter REMOTE_LOGIN_PASSWORDFILE is set to SHARED for multiple databases. There is no *`sid`* specific password file for multiple databases.

Access to a Database from a Remote PC

When there is an Oracle8 password file, networked PC users with DBA privileges can access this database as INTERNAL. Non-privileged users can connect to the database by invoking an Oracle application that uses the database. Privileged users who want to perform DBA functions on the database can enter the appropriate Server Manager command from their PC, adding the `dba` user password. For example:

```
SVRMGR> CONNECT INTERNAL/dba_password
```

To connect as OPERATOR, use the same command with the OPERATOR password.

Remote Authentication

Table 2–12 shows the `init`*`sid`*`.ora` parameters that control the behavior of remote connections through non-secure protocols.

Table 2–12 Parameters For Controlling Remote Connections

Parameter	Description
REMOTE_OS_AUTHENT	Enables or disables ops$ connection
OS_AUTHENT_PREFIX	Used by `OPS$` accounts
REMOTE_OS_ROLES	Enables or disables roles through remote connections

> **Note:** If REMOTE_OS_AUTHENT is set to true, users who are members of the `dba` group on the remote computer are able to connect as INTERNAL without a password.

User-Visible Effects of the Shutdown Mechanism

Clients connected to an Oracle instance while a shutdown takes place receive one of the following error messages upon subsequent SQL operations:

```
ORA-03113: end-of-file on communication channel
ORA-12571: TNS:packet writer failure
```

Administering Login Home Directories

To add or move login home directories without modifying programs that refer to them, you must do one of the following:

- refer to explicit path names in files designed to store them, for example: `/etc/passwd` and `/var/opt/oracle/oratab`
- refer to group memberships in the `/etc/group` file

You only need to record a path name in the central reference file because a user's home directory can be derived in either of the following ways:

- C shell and Korn shell users can use *~login* to refer to a user's home directory.
- Bourne shell users can construct a simple program to refer to a user's home directory. See the sample `lhd` script later in this section.

Similarly, group memberships are calculated from `/etc/group`. See Example 2-2.

Note: Local general-purpose utilities such as these should be stored in the `/var/opt/bin` directory.

Example 2-2 Sample lhd Script

```
#!/bin/sh
#
# lhd - print login home directory name for a given user
# SYNTAX
# lhd [login]
#
prog=‘basename $0‘
if [ $# -eq 0 ] ; then
    login=‘whoami‘

elif [ $# -eq 1 ] ; then
    login=$1
```

```
else
    echo "Usage: $prog login" >$2

    exit 2
fi
nawk -F: '$1==login {print $6}' login=$login /etc/passwd
```

Example 2-3 Sample grpx Script

```
#!/bin/sh
# grpx - print the list of users belonging to a given group
#
prog=`basename $0`
if [ $# -ne 1 ] ; then
    echo "Usage: $prog group" >&2
    exit 2
fi
g=$1
# calculate group id of g
gid=`nawk -F: '$1==g {print $3}' g=$g /etc/group`
# list users whose default group id is gid
u1=`nawk -F: '$4==gid {print $1}' gid=$gid /etc/passwd`
# list users who are recorded members of g
u2=`nawk -F: '$1==g {gsub(/,/," "); print $4}' g=$g /etc/group`
# remove duplicates from the union of the two lists
echo $u1 $u2 | tr " " "\012" | sort | uniq | tr "\012" " "
echo
```

Propagating a Skeleton .profile File

Example 2-4 shows how the administrator can propagate a skeleton `.profile` file to the home directory for each member of a group. If the membership list of the `clerk` group changes, the code does not require modification.

Example 2-4 Propagating a Skeleton .profile File

```
$ for u in `grpx clerk` ; do
> cp /etc/skel/.profile `lhd $u`
> done
```

Building and Running Demonstrations

This section describes how to load and run the demonstration programs installed with Oracle8.

Loading PL/SQL Demonstrations

PL/SQL includes a number of sample programs that you can load. Demonstration and message files are in the `rdbms` directory. Perform these steps with Oracle8 open and mounted:

1. Invoke Server Manager and connect with the user name and password SCOTT/TIGER:

   ```
   $ cd $ORACLE_HOME/plsql/demo
   $ svrmgrl
   SVRMGR > CONNECT SCOTT/TIGER
   ```

2. To load the demonstrations, run the `exampbld.sql` script from Server Manager.

   ```
   VRMGR > @exampbld
   ```

   Note: Build the demonstrations under any Oracle account with sufficient permissions. Run the demonstrations under the same account that you used to build them.

Running PL/SQL Demonstrations

The following PL/SQL demonstrations are available:

```
examp1.sql     examp5.sql      examp11.sql    sample1.sql
examp2.sql     examp6.sql      examp12.sql    sample2.sql
examp3.sql     examp7.sql      examp13.sql    sample3.sql
examp4.sql     examp8.sql      examp14.sql    sample4.sql
extproc.sql
```

The following precompiler demonstrations are available:

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

To run the kernel PL/SQL demonstrations, start SQL*Plus to connect to the kernel, using the same user name and password that you used to create the demonstrations. Start the demonstration by typing an at sign (@) or the word START before the demonstration name. For example, to start the examp1 demonstration, enter:

```
$ sqlplus SCOTT/TIGER
SQLPLUS > @examp1
```

To build the precompiler PL/SQL demonstrations, enter:

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

If you want to build a single demonstration, enter its name as the argument in the make command. For example, to make the examp9.pc executable, enter:

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

To start the examp9 demonstration from your current shell, enter:

```
$ examp9
```

In order to run the extproc demo, you must:

1. Add the following line to the file, tnsnames.ora:

   ```
   (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=plsff))(CONNECT_
   DATA=(SID=extproc)))
   ```

2. Add the following line to the file, listener.ora:

   ```
   SC=(SID_NAME=extproc)(ORACLE_HOME=/vobs/oracle)(PROGRAM=extproc))
   ```

3. Start Server Manager and enter the following commands:

   ```
   SVRMGR> CONNECT SCOTT/TIGER
   CONNECTED.
   SVRMGR> CONNECT SYSTEM/MANAGER
   CONNECTED.
   SVRMGR> GRANT CREATE LIBRARY to scott;
   STATEMENT PROCESSED.
   SVRMGR> CONNECT SCOTT/TIGER
   CONNECTED.
   SVRMGR> CREATE LIBRARY DEMOLIB AS
   '$ORACLE_HOME/plsql/demo/extproc.so';
   STATEMENT PROCESSED.
   ```

4. To run the tests, enter:

```
SVRMGR> CONNECT SCOTT/TIGER
CONNECTED.
SVRMGR> @extproc
```

> **See Also:** For more information about running the `extproc` demonstration, refer to the *PL/SQL User's Guide and Reference*.

SQL*Loader Demonstrations

SQL*Loader demonstrations require that:

- The user `SCOTT/TIGER` has CONNECT and RESOURCE privileges
- The EMP and DEPT tables exist and are empty

To create and run a demonstration:

1. Connect to the database as the user name and password `SCOTT/TIGER` from Server Manager (line mode).
2. Run the `ulcase`*n*`.sql` script that corresponds to the demonstration that you want to run.
3. As `SCOTT/TIGER`, start the demonstration from the command line:

   ```
   $ sqlldr scott/tiger ulcasen
   ```

As `SCOTT/TIGER`, run the SQL*Loader demonstrations in the following order:

`ulcase1` Follow steps 1 through 3.

`ulcase3` Follow steps 1 through 3.

`ulcase4` Follow steps 1 through 3.

`ulcase5` Run the `ulcase*.sql` script as SCOTT/TIGER, then enter the following at the command line:

```
$ SQLLDR SCOTT/TIGER ULCASE*
```

`ulcase2` Start the demonstration (you do not have to run the `ulcase2.sql` script).

`ulcase6` Run the `ulcase6.sql` script as `SCOTT/TIGER`, then enter the following at the command line:

```
$ SQLLDR SCOTT/TIGER ULCASE1 DIRECT=TRUE
```

`ulcase7` Run the `ulcase6.sql` script as `SCOTT/TIGER`, then enter the following at the command line:

```
$ SQLLDR SCOTT/TIGER ULCASE7
```

Administering SQL*Loader

This section describes SQL*Loader file processing options and newline characters in fixed length records.

Oracle8 incorporates SQL*Loader functionality. Demonstration and message files are in the `rdbms` directory.

File Processing Option

The SQL*Loader release 1.1 control file includes the following additional file processing option strings:

```
["STR" | "FIX N" | "VAR N" ]
```

where;

`STR` (default) specifies a stream of records, each terminated by a newline character, which are read in one record at a time

`FIX` indicates that the file consists of fixed-length records, each of which is *n* bytes long, where *n* is an integer value

`VAR` indicates that the file consists of variable-length records, each of which is *n* bytes long, where *n* is an integer value specified in the first five characters of the record

If you do not select a file processing option, the information is processed by default as a stream of records (`STR`). You might find that `FIX` mode yields faster performance than the default `STR` mode because it does not need to scan for record terminators.

Newlines in Fixed Length Records

When using the FIX option to read a file containing fixed-length records, where each record is terminated by a newline, include the length of the newline (one character) when specifying the record length to SQL *Loader.

For example, specify `FIX 4` instead of `FIX 3` to account for the additional newline character when reading read the following file:

```
AAA newline
BBB newline
CCC newline
```

If you do not terminate the last record in a file of fixed records with a newline character, do not terminate the other records with a newline character either. Similarly, if you terminate the last record with a newline, terminate all records with a newline.

> **CAUTION: Certain text editors, such as `vi`, automatically terminate the last record of a file with a newline character. This leads to inconsistencies if the other records in the file are not terminated with newline characters.**

Removing Newlines

Use the `POSITION(X:Y)` function in the control file to discard the newlines from fixed length records rather than loading them. To do this, enter the following in your control file:

```
LOAD DATA
INFILE xyz.dat "FIX 4"
INTO TABLE ABC
(DEPT POSITION(01:03) CHAR )
```

When this is done, newlines are discarded because they are in the fourth position in each fixed-length record.

Oracle Security Server

For information on the Oracle Security Server see the *Oracle Security Server Guide.*

Database Limits

Table 2–13 lists the maximum and default values for parameters in a CREATE DATABASE or CREATE CONTROLFILE statement.

Table 2–13 Determining the Size of Control Files

Parameter	Default Value	Maximum Value
MAXDATAFILES	30	1022
MAXINSTANCES	1	63
MAXLOGFILES	16	255
MAXLOGMEMBERS	2	5
MAXLOGHISTORY	100	65534

3

Tuning Oracle8 on SGI IRIX

This chapter describes how to configure your Oracle8 installation to optimize its performance. It contains the following sections:

- The Importance of Tuning
- Tools for Monitoring Database Performance
- SQL Scripts
- Tuning Memory Management
- Tuning Disk I/O
- Monitoring Disk Performance
- Tuning CPU Usage
- Tune UNIX Kernel Parameters
- Tuning Block Size and File Size
- Tuning the Buffer Cache Size
- Performance Tuning Specific to IRIX Systems
- Using Trace and Alert Files
- Raw Devices

The Importance of Tuning

Oracle8 is a highly optimizable software product. Frequent tuning optimizes system performance and prevents data bottlenecks.

Before Tuning the System

Before tuning the system, observe its normal behavior using the tools described in "Tools for Monitoring Database Performance" in the next section.

Tools for Monitoring Database Performance

IRIX provides performance monitoring tools that you can use 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, and context switching for the entire system.

> **See Also:** For information on IRIX tools, see the operating system documentation.

sar

Use the `sar` command to monitor swapping, paging, disk, and CPU activity, depending on the switches that you supply with the command. The following statement displays a summary of paging activity ten times, at 10 second intervals:

```
$ sar -p 10 10
```

Sample output from the `sar -p` command is shown in Example 3–1.

Example 3–1 Output from the sar -p Command

```
14:14:55  atch/s  pgin/s ppgin/s  pflt/s  vflt/s slock/s
14:15:05    0.00    0.00    0.00    0.60    1.00    0.00T
14:15:15    0.00    0.00    0.00    0.10    0.60    0.00
14:15:25    0.00    0.00    0.00    0.00    0.00    0.00
14:15:35    0.00    0.00    0.00    0.00    0.00    0.00
14:15:45    0.00    0.00    0.00    0.00    0.00    0.00
14:15:55    0.00    0.00    0.00    0.00    0.00    0.00
14:16:05    0.00    0.00    0.00    0.00    0.00    0.00
14:16:15    0.00    0.00    0.00    0.00    0.00    0.00
14:16:25    0.00    0.00    0.00    0.00    0.00    0.00
14:16:35    0.00    0.00    0.00    0.00    0.00    0.00

Average     0.00    0.00    0.00    0.07    0.16    0.00
```

swap

The `swap -l` command reports information about swap space usage. A shortage of swap space can cause slow response times or even cause the system to hang. Example 3–2 contains the sample output from the `swap -l` command.

Example 3–2 Sample Output From a swap -l Command

```
swapfile             dev    swaplo   blocks    free
/dev/dsk/dks3d0s1    32,25       8   197592   162163
```

SQL Scripts

The `utlbstat` and `utlestat` SQL scripts are used to monitor Oracle database performance and tune the Shared Global Area (SGA) data structures. For information about these scripts, see the *Oracle8 Tuning* guide. On IRIX systems, the scripts are located in the `$ORACLE_HOME/rdbms/admin/` directory.

Tuning Memory Management

Start the memory tuning process by tuning paging and swapping space to determine how much memory is available.

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

Allocate Sufficient Swap Space

Try to minimize swapping because it causes significant UNIX overhead. Use `sar -w` on IRIX to check for swapping.

If your system is swapping and you need to conserve memory:

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

Start with a swap-space two to four times the size of your system's random access memory (RAM). Use a higher value if you plan to use Oracle Developer or Oracle Applications. Monitor the use of swap space and increase it as necessary.

Control Paging

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

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

Use `sar -p` to monitor paging. The following columns from `sar -p` output are important:

`vflt/s` indicates the number of address translation page faults. Address translation faults occur when a process references a valid page not in memory.

`rclm/s` indicates the number of valid pages that have been reclaimed and added to the free list by page-out activity. This value should be zero.

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

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

Configure Shared Memory

The optimal value of SHMMAX for Oracle8 on SGI IRIX is 512 MB.

Lock the SGA in Physical Memory

The primary function of the SGA is to cache database information. If the SGA begins paging to disk, caching becomes an overhead rather than a benefit. If you have installed Oracle8 on SGI IRIX and have installed the IRIX-specific enhancements, you can use the LOCK_SGA parameter to lock the memory pages associated with the SGA.

Such locking prevents paging and helps asynchronous I/O on raw files to work efficiently. Locked memory is not available for use by other applications.

This option should be used only if there is enough physical memory on the system to support the Oracle instance, the applications, and other users. If the amount of physical memory on the system is insufficient, performance degradation might occur due to increased memory paging or swapping.

Refer to the `mpin(2)` man page for further information.

Tuning Disk I/O

I/O bottlenecks are the easiest performance problems to identify. Balance I/O evenly across all available disks to reduce disk access times. For smaller databases and those not using the Parallel Query option, ensure that different datafiles and tablespaces are distributed across the available disks.

Tune the Database Writer to Increase Write Bandwidth

Oracle provides asynchronous I/O and I/O slaves as solutions to prevent database writer (DBWR) activity from becoming a bottleneck.

Asynchronous I/O

Asynchronous I/O allows processes to proceed with the next operation without having to wait after issuing a write and therefore improves system performance by minimizing idle time. IRIX supports asynchronous I/O to both raw and filesystem datafiles.

I/O Slaves

I/O slaves are specialized processes whose only function is to perform I/O. They are new with Oracle8, and replace Multiple DBWRs. They are a generalization of Multiple DBWRs and can also be deployed by other processes and can operate whether or not asynchronous I/O is available. I/O slaves include a new set of initialization parameters which allow a degree of control over the way they operate.

Table 3–1 lists the initialization parameters that control the operation of asychronous I/O and I/O slaves.

Table 3–1 Initialization Parameters for I/O Slaves

Parameter	Range of Values	Default Value
DISK_ASYNCH_IO	TRUE/FALSE	FALSE
TAPE_ASYNCH_IO	TRUE/FALSE	TRUE
BACKUP_DISK_IO_SLAVES	TRUE/FALSE	FALSE
BACKUP_TAPE_IO_SLAVES	TRUE/FALSE	FALSE
DBWR_IO_SLAVES	0 - 999	0
LGWR_IO_SLAVES	0 - 999	0
ARCH_IO_SLAVES	0 - 999	0
DB_WRITER_PROCESSES	1-10	1

There might be times when the use of asynchronous I/O is not desirable or not possible. The first two parameters in Table 3–1, DISK_ASYNCH_IO and TAPE_ASYNCH_IO, allow asynchronous I/O to be switched off respectively for disk and tape devices. Because the number of I/O slaves for each process type defaults to zero, no I/O slaves will be deployed unless specifically set.

Set the DBWR_IO_SLAVES parameter to greater than 0 only if ASYNC I/O (that is, DISK_ASYNCH_IO, or TAPE_ASYNCH_IO) has been disabled, otherwise DBWR becomes a bottleneck. For LGWR_IO_SLAVES, Oracle Corporation recommends that you do not deploy more than 9 slaves.

DB_WRITER_PROCESSES replaces the Oracle7 DB_WRITERS parameter, and specifies the initial number of database writer processes for an instance. If you use DBWR_IO_SLAVES, only one database writer process is used, regardless of the setting for DB_WRITER_PROCESSES.

Check Size of Disk I/O Request Queues

A request queue shows how long the I/O operations on a particular disk device must wait to be serviced. Request queues grow longer if there is a high volume of I/Os to that disk or if there are many I/O operations with long average seek times. Ideally, I/O request queues should be at zero or near it. The “Resp Time” field in the File I/O Monitor in Server Manager shows how long requests are waiting.

Use More Database Buffers

If your system is I/O bound, increase the number of database buffers to cache more data and reduce I/O. Continue increasing the number of buffers (and the hit ratio) as long as it does not increase paging.

Choose the Appropriate File System Type

SGI IRIX allows a choice of file systems. File systems have different characteristics, and the techniques they use to access data can have a substantial impact on database performance. Typical file system choices are:

- The EFS file system
- The XFS file system
- Data access through an XLV device
- Data access through raw device drivers

> **Note:** Oracle Corporation discourages the use of the EFS file system. The EFS file system might not be supported in future versions of Oracle software.

Use Raw Partitions and Devices (I/O Bound Systems)

Using raw partitions instead of a file system can improve performance because the database writer bypasses the UNIX buffer cache and eliminates the file system overhead. This results in fewer instructions per I/O.

> **Note:** This is a global operation. It can adversely affect other applications on the same system.

When Disk I/O Optimizations Fail

When disk I/O optimizations fail to eliminate I/O bottlenecks, you might need to move some applications to another system or add more disk drives and controllers to your system.

Monitoring Disk Performance

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

Table 3–2 lists important `sar -b` columns for disk performance.

Table 3–2 Output of sar-b Significant for Monitoring Disk Performance

Field	Value Shown	Database Type
`bread/s`	Blocks read	File system
`bwrit/s`	Blocks written	File system
`pread/s`	Partition reads	Raw device
`pwrit/s`	Partition writes	Raw device

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

Key indicators are:

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

Disk Performance Issues

Oracle block sizes should either match disk block sizes, or be a multiple of disk block sizes.

If possible, perform a file system check on the partition before using it for database files, then make a new file system to ensure that it is clean and unfragmented. Distribute disk I/O as evenly as possible and separate log files from database files.

Tuning CPU Usage

The following sections describe how to tune the CPU.

Keep All Oracle Users/Processes at the Same Priority

Oracle is designed to operate with all users and background processes operating at the same priority level. Changing priorities causes unexpected effects on contention and response times.

For example, if the log writer process (LGWR) gets a low priority, it is not executed frequently enough and LGWR becomes a bottleneck. On the other hand, if LGWR has a high priority, user processes can suffer poor response time.

Use Processor Affinity/Binding on Multi-Processor Systems

In a multi-processor environment, use processor affinity / binding if it is available on your system. Processor binding prevents a process from migrating from one CPU to another, allowing the information in the CPU cache to be better utilized. You can bind a server shadow process to make use of the cache as it is always active, and let background processes flow between CPUs. Some platforms employ process binding automatically.

Use a Client/Server Configuration

If your system is CPU-bound, move applications to a separate system to reduce the load on the CPU. For example, you can off-load foreground processes such as Oracle Forms to a client system to free CPU cycles on the database server system.

Use the Post-Wait Driver

Oracle processes usually use semaphores to coordinate access to shared resources. If a shared resource is locked, a process suspends and waits for the resource to become available.

One way to improve shared resource coordination is to use a post-wait driver instead of semaphores, if it is available on your system. A post-wait driver is a faster, less expensive synchronization mechanism than a semaphore. The Oracle Post-Wait driver implements an optimized mechanism of inter-process communication, without the overhead of signal handlers or semaphores. It improves performance for Oracle8.

Use Single-Task Linking for Large Exports/Imports and SQL*Loader Jobs

If you need to transfer large amounts of data between the user and Oracle8 (for example, using `export/import`), it is efficient to use single-task architecture. To make the single-task import (`impst`), export (`expst`), and SQL*Loader (`sqlldrst`) executables, use the `ins.rdbms.mk` makefile, which can be found in the `$ORACLE_HOME/rdbms/lib` directory.

The following example makes the `impst`, `expst`, and `sqlldrst` executables:

```
% cd $ORACLE_HOME/rdbms/lib
% make -f ins_rdbms.mk expst impst sqlldrst
```

> **Note:** Linking Oracle executables as a single-task allows a user process to directly accesses the entire SGA. In addition, running single-task requires more memory because the `oracle` executable text is no longer shared between the front-end and background processes.

Tune UNIX Kernel Parameters

You can improve performance by keeping the UNIX kernel as small as possible. The UNIX kernel typically pre-allocates physical RAM, leaving less memory available for other processes, such as `oracle`.

Traditionally, kernel parameters such as NBUF, NFILE, and NOFILES were used to adjust kernel size. However, most UNIX implementations dynamically adjust those parameters at runtime, even though they are present in the UNIX configuration file.

Look for memory mapped video drivers, networking drivers, and disk drivers. They can often be de-installed, yielding more memory for use by other processes.

> **Note:** Remember to make a backup copy of your UNIX kernel. See your hardware vendor documentation for more information.

Tuning Block Size and File Size

This section describes how you can improve the performance of Oracle8 by optimizing the size of Oracle blocks for the files in your database.

Caution: To change block size, you must create a new database. To determine the most efficient configuration, experiment with block size before transferring your data to the new database.

Block Size and File Size

Although data storage space is often measured in megabytes, the UNIX operating system and Oracle8 each perform input and output in units of data storage called *blocks*. The size of the operating system blocks is not necessarily equal to Oracle blocks.

The Oracle8 block size can be set when you create a database by changing the DB_BLOCK_SIZE parameter in the `init`*`sid`*`.ora` file.

Changing the Oracle block size can change database performance, depending on the disk hardware, file system, and application. The default block size is adequate for most circumstances, but some performance benefits can be gained by modifying it. But remember, to change block size, you must create a new database. To determine the most efficient configuration, experiment with block size before transferring your data to the new database.

Specifying Oracle Block Size

On SGI IRIX, the default Oracle block size is 2KB and the maximum block size is 16KB.

You can set the actual block size to any multiple of 2KB up to 16KB.

The optimal block size is typically the default, but varies with the applications. To create a database with a different Oracle block size, add the following line to the `init`*`sid`*`.ora` file:

```
db_block_size=new_block_size
```

Associating Data Blocks with Instances

If you have multiple Oracle instances accessing a single Oracle database, you can use free lists. This allows transactions running on separate instances to insert and update data in the same table concurrently.

When an insert or update command must locate free space in a table, Oracle8 searches one of the free lists in the set associated with the instance running that transaction. If the free list does not contain a block with sufficient space, Oracle8 searches the master free list, not one of the other free lists for that instance.

If the master free list does not contain sufficient space, Oracle8 allocates a new extent, if possible, and adds its space to the master free list. It is important to pre-allocate extents to tables; otherwise, the free space in an extent allocated by Oracle8 is available to all instances and you lose the performance advantages of partitioning data between instances.

To prevent Oracle8 from allocating new extents for a table, set MAXEXTENTS to the number of pre-allocated extents plus MINEXTENTS.

If Oracle8 cannot find sufficient space on the master free list and cannot allocate a new extent, it displays the following error message:

`ORA-01547: failed to allocate extent of size` *num* `in tablespace` *name*

Oracle8 allows explicit allocation of new space to a table and the ability to specify a database file from which to take the new space. Use the options of the CREATE TABLE and ALTER TABLE SQL statements to associate free space with particular instances:

- CREATE TABLE has two storage options. The FREELIST GROUPS option specifies the number of free list sets for the table. The FREELISTS option specifies the number of free lists per set.
- ALTER TABLE has the option ALLOCATE EXTENT, which allocates and associates space with a particular set of free lists.

In a multi-instance server configuration, MAXINSTANCES for the database could be many times larger than FREELIST GROUPS for a table, so that many instances share one set of free lists.

> **See Also:** For more information about the options of the CREATE TABLE and ALTER TABLE commands, refer to the O*racle8 SQL Reference* guide.

Table Striping

Table striping is the process of dividing the data for a large table into small portions and storing these portions in separate data files on separate disks. Striping enables multiple processes to access different portions of the table concurrently without disk contention. Striping is particularly helpful in optimizing random access to tables with many rows.

> **See Also:** For more information about the ALLOCATE EXTENT DATAFILE parameter of the ALTER TABLE command, refer to the O*racle8 SQL Reference* guide.

Tuning the Buffer Cache Size

To take full advantage of raw devices, adjust the size of the Oracle8 buffer cache and, if memory is limited, the IRIX buffer cache.

The buffer cache is provided by the operating system. It holds blocks of data in memory while they are being transferred between memory and disk.

The Oracle8 buffer cache is the area in memory that stores the Oracle database buffers. Since Oracle8 can use raw devices, it does not need to use the IRIX buffer cache.

If you decide to change to raw devices, you must increase the size of the Oracle8 buffer cache. If the amount of memory on the system is limited, reduce the size of the operating system's buffer cache correspondingly.

Use the `sar` command to determine which buffer caches you need to increase or decrease. The options of the `sar` command are shown in Table 3–3.

Table 3–3 Useful sar Command options

Option	Description
`-b`	Reports the IRIX buffer cache activity
`-w`	Reports the IRIX swapping activity
`-u`	Reports CPU utilization
`-r`	Reports memory utilization
`-p`	Reports the IRIX paging activity

Increase the Oracle8 cache size until the increase causes the cache hit ratio to increase.

If swapping or paging activity becomes very high, decrease the cache size.

Performance Tuning Specific to IRIX Systems

The following sections describe how to tune Oracle8 on SGI IRIX systems.

IRIX Processor Affinity

You can specify which processor (CPU) handles each Oracle8 process on your system by setting the IRIX_CPU_AFFINITY initialization parameter.

For more information about using IRIX Processor Affinity, see “Tuning Oracle8 on IRIX Systems” on page 2-21.

IRIX Scheduler Options

You can specify certain IRIX Scheduler options by setting the IRIX_SCHEDULER initialization parameter.

You can affect Oracle8 processes by:

- Setting or removing non-degrading priorities
- Changing their `nice` values
- Changing their time slices

Refer to the `schedctl(2)` man page for details on the restrictions and uses of this option.

For more information about using the IRIX Scheduler, see "Tuning Oracle8 on IRIX Systems" on page 2-21.

Using Trace and Alert Files

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

Trace File Names

The format of a trace file name is *`processname_sid_unixpid`*`.trc`**,** where:

Table 3–4 Format Key to Process Name

`processname`	is a three- or four-character process name showing which Oracle8 process the trace file is from (for example, PMON, DBWR, ORA, or RECO)
`sid`	is the instance system identifier
`unixpid`	is the UNIX process ID number
`.trc`	is a file name extension appended to all trace file names

A sample trace file name is `lgwr_TEST_1237.trc`.

Alert Files

The `alert_`*`sid`*`.log` file is associated with a database and is located in the directory specified by the `init`*`sid`*`.ora` parameter BACKGROUND_DUMP_DEST. The default value is `$ORACLE_HOME/rdbms/log`.

Raw Devices

This section describes the use of raw devices on Oracle8.

Disadvantages of Raw Devices

Raw devices have the following disadvantages when used on SGI IRIX:

- They might not solve problems with ULIMIT that can arise when exporting tables larger than a megabyte (such as another disk partition).
- When raw devices and operating system files are mixed within an Oracle8 database, the operating system files must still be within the value of the ULIMIT parameter.
- They might not solve problems with ULIMIT that can arise when reading the contents of the Oracle distribution media onto the disk.
- Small client systems usually cannot use sufficiently large raw device partitions. Disk partitions usually come in odd sizes that might hinder good database architecture.
- If a particular disk drive has intense I/O activity and performance would benefit from movement of an Oracle data file to another drive, it is likely that no acceptably sized section exists on a drive with less I/O activity. Moving data files around, a common advantage of UNIX, might not be possible with raw devices.
- Adding space to a tablespace can be a difficult process in a raw device environment. Occasionally, all raw partitions are assigned data files at initial configuration time, leaving no raw storage to accommodate normal tablespace growth.

Criteria for Using Raw Devices

Consider the availability of raw disk partitions when deciding on raw devices.

Raw Disk Partition Availability

Use raw devices for Oracle files if your site has at least as many raw disk partitions as Oracle tablespaces.

If the raw disk partitions are already formatted, match tablespace size to partition size as closely as possible to avoid wasting space.

Guidelines for Using Raw Devices

When creating raw disk partitions, observe these guidelines:

- Three partitions for the log files of each instance
- One partition each for the following datafiles: SYSTEM, ROLLBACK, TEMP, USERS, TOOLS
- At least three partitions for data files

Configuration Planning

With logical volumes, you can create logical disks based on raw partition availability, because logical disks can be moved on more than one disk. The disk drives do not have to be reformatted to obtain logical disk sizes.

Dynamic Performance Tuning

You can optimize disk performance when the database is online by moving hot spots to cooler drives. Most hardware vendors who provide the logical disk facility also provide a graphical user interface that can be used for tuning.

Mirroring and Online Disk Replacement

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

Setting Up Raw Devices

> **CAUTION: Do not attempt to set up raw devices without the help of an experienced system administrator and specific knowledge about the system you are using.**

To set up raw devices on your system:

1. Make sure that the partitions you are adding are on a shared disk.
2. Determine the names of the free disk partitions.

   A free partition is one that is not used for an SGI IRIX file system. That means that the partition follows these restrictions:

   - It is not listed when you execute the `/sbin/mount` or `/etc/mount` command.
   - It is not in use as a swap device.
   - It does not overlap a swap partition.
   - It is not in use by other SGI IRIX applications (for example, other instances of Oracle).
   - It does not overlap an XFS or EFS filesystem or a space used by XLV.

   To find out whether a partition is free, obtain a complete map of the starting locations and sizes of the partitions on the device and check for free space. Note that some partitions might contain file systems that are currently not mounted and are not listed in the `/sbin/mount` or `/etc/mount` output.

   > **Attention:** Make sure that the partition does *not* start at Cylinder 0.

3. Set up the raw device for use by the Oracle8 Server.

   a. Verify that the disk is partitioned. If not, use the operating system `format` utility to partition it.

   b. Ensure that the partition is owned by the *`oracle`* software owner. If necessary, use `chown` to change its ownership. For example:

   ```
   $ chown oracle /dev/rdsk/dks1d2s7
   ```

c. Use chmod to make the partition accessible only by the *oracle* software owner. For example:

```
$ chmod 600 /dev/rdsk/dks1d2s7
```

4. Create or add the new partition to a new database.

From Server Manager, use the CREATE DATABASE SQL statement to create the database using the specified raw partition.

Note: The size of an Oracle datafile created in a raw partition must be at least two Oracle block sizes smaller than the size of the raw partition.

If you want to add the partition to a tablespace in an existing Oracle database instead, enter:

```
$ svrmgrl
SVRMGR> alter tablespace tablespace_name add datafile
'/dev/rdsk/dks1d2s7' size 10000K reuse;
```

You can use the same procedure to set up a raw device for the redo log files.

4

Administering SQL*Plus on SGI IRIX Systems

This chapter describes the administration tasks that you must perform to enable users to run SQL*Plus on SGI IRIX. It contains the following sections:

- Administering SQL*Plus
- Using SQL*Plus
- Restrictions

Administering SQL*Plus

When you run SQL*Plus, it executes the SQL statements and SQL*Plus commands in the files described in the following sections.

Setup Files

The setup files for SQL*Plus are `glogin.sql`, the global setup file that defines the site profile, and `login.sql`, which defines the user profile. The `glogin.sql` and `login.sql` files contain SQL statements or SQL*Plus commands that you choose to execute at the beginning of each SQL*Plus session. When you start SQL*Plus, `glogin.sql` is read first, followed by `login.sql`.

Site Profile

The site profile file is `$ORACLE_HOME/sqlplus/admin/glogin.sql`. SQL*Plus executes this command file whenever any user starts SQL*Plus and SQL*Plus establishes the Oracle connection. The default site profile is placed in the `$ORACLE_HOME/sqlplus/admin` directory whenever SQL*Plus is installed. If a site profile already exists, it is overwritten. An existing site profile is deleted whenever SQL*Plus is de-installed.

User Profile

The user profile file is `login.sql`. SQL*Plus attempts to execute this command file whenever any user starts SQL*Plus and SQL*Plus establishes the Oracle connection. The user profile is run after the site profile. SQL*Plus always searches the current directory for the user profile. The environment variable SQLPATH can be set to a colon-separated list of directories that SQL*Plus searches in the order specified.

For example, if the current directory is `/u02/oracle` and SQLPATH is set to `/home:/home/oracle:/u01/oracle`, SQL*Plus will first look for `login.sql` in the current directory `/u02/oracle`. If it does not find it there, SQL*Plus then looks in the `/home`, `/home/oracle` directory and the `/u01/oracle` directory respectively. SQL*Plus then runs the first `login.sql` script it finds. Because `login.sql` is run last, options set in `login.sql` over-ride those set in `glogin.sql`.

Here is a sample `login.sql` file:

```
set echo off
set feedback 4
set pause on
set pause "PLEASE PRESS RETURN TO CONTINUE"
set message on
set echo on
```

The PRODUCT_USER_PROFILE Table

You can run the `$ORACLE_HOME/sqlplus/admin/pupbld.sql` SQL script as the SYSTEM user to create the Product and User Profile tables.

You can also run the `$ORACLE_HOME/sqlplus/admin/pupbld.sql` script by running the shell script, `$ORACLE_HOME/bin/pupbld`. To use this script, you must first set the ORACLE_HOME and SYSTEM_PASS environment variables. The SYSTEM_PASS environment variable must be set to the SYSTEM's user name and password. For example:

```
% setenv SYSTEM_PASS SYSTEM/manager
% pupbld

Installing product user profile tables...

Product user profile tables installed.
```

The installer runs the `pupbld.sql` script while installing SQL*Plus only if Create Database Objects is selected.

Enabling the SQL*Plus Demonstrations

The `$ORACLE_HOME/sqlplus/demo` directory contains demonstration files that are installed automatically by the Oracle installer.

Installing the SQL*Plus Demonstrations

The following sections describe the installation options available for the SQL*Plus demonstrations.

Default Install

If you select Default Install and Create Database Objects, the user SCOTT and the demonstration tables are created automatically.

Custom Install

When installing SQL*Plus using Custom Install, if you select Create Database Objects and answer 'Yes' to the prompt “Would you like to load the SQL*Plus Demo Tables?”, the Installer creates the user SCOTT with the password TIGER and creates the demonstration tables.

Creating Demonstration Tables Manually

Run the `$ORACLE_HOME/sqlplus/demo/demobld.sql` SQL script to create the demonstration tables. You can run the script in SQL*Plus as any user to create the demonstration tables in that schema. For example:

```
% sqlplus SCOTT/TIGER
SQL> @?/SQLPLUS/DEMO/DEMOBLD.SQL
```

You can also run the `$ORACLE_HOME/sqlplus/demo/demobld.sq` script using the shell script `$ORACLE_HOME/bin/demobld` as follows:

```
% demobld SCOTT TIGER
```

Deleting Demonstration Tables

Run the $ORACLE_HOME/sqlplus/demo/demodrop.sql SQL script to drop the demonstration tables. You can run the demodrop.sql script in SQL*Plus as any user to drop the demonstration tables from that user's schema. For example:

```
% sqlplus SCOTT/TIGER
SQL> @?/SQLPLUS/DEMO/DEMODROP.SQL
```

$ORACLE_HOME/sqlplus/demo/demodrop.sql can also be run using the shell script $ORACLE_HOME/bin/demodrop, as follows:

```
% DEMODROP SCOTT TIGER
```

> **Note:** Both the demobld.sql and demodrop.sql SQL scripts drop the EMP, DEPT, BONUS, SALGRADE, and DUMMY tables. You must ensure that a table does not exist with the same name in the desired schema prior to running either script or the table data will be lost.

Enabling SQL*Plus Help

If you select Default Install and Create Database Objects in your installer session, the SQL*Plus Help facility is installed automatically with the software. This section describes how to install SQL*Plus Help when you are not using the Default Install method.

Custom Install

When you install SQL*Plus, if you select Create Database Objects and answer 'Yes' to the prompt "Would you like to load the SQL*Plus Help Facility?", the Installer creates the Help Facility.

Installing the Help Facility Manually

You can install the Help Facility manually using the shell script $ORACLE_HOME/bin/helpins. To use this script, you must set the ORACLE_HOME and SYSTEM_PASS environment variables. Set SYSTEM_PASS to SYSTEM's user name and password. For example:

```
$ setenv SYSTEM_PASS SYSTEM/MANAGER
$ helpins

SQL*Loader: Release 8.0.6.0.0 - Production

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Commit point reached - logical record count 828

SQL*Loader: Release 8.0.6.0.0 - Production

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Commit point reached - logical record count 1024
Commit point reached - logical record count 1207

SQL*Loader: Release 8.0.6.0.0 - Production

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Commit point reached - logical record count 1024
Commit point reached - logical record count 1304
Commit point reached - logical record count 2328
Commit point reached - logical record count 2724
Commit point reached - logical record count 2835
```

See Also: Refer to the *SQL*Plus User's Guide and Reference*, and the $ORACLE_HOME/sqlplus/doc/release.doc README file.

Using SQL*Plus

The following section describes how to use some features of SQL*Plus with Oracle8.

Specifying a System Editor for SQL*Plus

You can start the default editor for your system, (such as `ed, emacs, ned`, or `vi`) by entering either of the commands, `ed` or `edit`, at the SQL*Plus prompt. The directory containing the default editor must be included in your PATH variable. Your PATH variable must include the directory of the editor.

The global default editor is usually set by the DBA in `glogin.sql` using the SQL*Plus `_editor` option. Override this setting by specifying an editor in the `login.sql` script. Both files are read by SQL*Plus at startup and the local file takes precedence. The user can set the `_editor` option during a SQL*Plus session to override the setting in either file.

If the `_editor` option is not set, the EDITOR and VISUAL environment variables specify the SQL*Plus editor. These variables are not set in `glogin.sql` or `login.sql`. They are set in a user startup file, or at the system prompt. If both are set, the EDITOR variable is used.

How SQL*Plus Selects the Default Editor

The following table lists, in order, the locations that SQL*Plus searches for the default editor and where it takes the setting from in each location:

Location	Setting taken from
SQL*Plus session	`_editor` option
`login.sql`	`_editor` option
`glogin.sql`	`_editor` option
UNIX environment	EDITOR environment variable
UNIX environment	VISUAL environment variable

If none of these values is set, SQL*Plus uses `ed`.

Setting the _editor option

Set the SQL*Plus `_editor` option by adding the following line to the `login.sql` file where `editor_name` is a UNIX editor, such as `vi` or `ed`:

```
define _editor=editor_name
```

Using Environment Variables to Specify an Editor

For the Bourne or Korn shell, specify the default editor in the EDITOR or VISUAL environment variable by entering:

```
$ EDITOR=editor_name; export EDITOR
```

For the C shell, set the default editor with an environment variable by entering:

```
% setenv EDITOR editor_name
```

Default Settings

If you call the system editor, the current SQL buffer is placed in the edit buffer and all statements available to the editor can change the SQL statement. SQL*Plus uses the `afiedt.buf` temporary file. When you exit the editor, the changed SQL buffer is returned to SQL*Plus.

Running Operating System Commands from SQL*Plus

The HOST command or an exclamation point ! as the first character after the SQL*Plus prompt indicates subsequent characters are passed to a sub-shell. The SHELL environment variable sets the shell used to execute operating system commands. The default shell is `/bin/sh(sh)`. If the shell cannot be executed, an error message is displayed.

You can perform operating system commands without leaving SQL*Plus by entering the `HOST` or ! commands.

For example, to enter one command, enter:

```
SQL>! command
```

In this example *command* represents the operating system command you want to execute. After the command has executed, control is returned to SQL*Plus.

To execute more than one operating system command, press Enter after the ! or HOST command.

Interrupting SQL*Plus

While running SQL*Plus:

- You can stop the scrolling record display and terminate a SQL statement by pressing [Ctrl]+[c] on BSD systems or [Delete] on System V systems.
- If you are at the SQL*Plus prompt, pressing [Interrupt] displays another SQL*Plus prompt.

Using the SPOOL Command

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

For example:

```
SQL> SPOOL query.lis
```

Restrictions

The following restrictions might apply to the use of SQL*Plus on your system.

COPY Command

The COPY command in SQL*Plus is supported without restrictions on different systems running the same version of the operating system.

COPY can also work between systems running different versions of the operating system. If COPY fails, test the connection using `rcp` or `ftp`. Restrictions in vendor-supplied networking software might prevent `rcp`, `ftp`, or COPY from functioning properly between systems.

Note: The `rlogin` command does not send or receive large packets of data and is not an adequate test for connections.

If COPY does not function between systems, create a database link to the system and user ID indicating the table you want to copy. To do this, enter:

```
SQL> create table newtable as (SELECT * FROM table@database_link_name)
```

This selects the rows and columns from the original table on the remote system and enters them in the new table on the local system.

Resizing Windows

The default value for SQL*Plus LINESIZE is 80 and for PAGESIZE is 25. These variables do not automatically adjust for window size.

Return Codes

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

5

Using Oracle Precompilers and the Oracle Call Interface

This chapter describes the administration tasks that you must perform to enable users to run Oracle precompilers on SGI IRIX systems. It contains the following sections:

- Overview of Oracle Precompilers
- Pro*C/C++
- Pro*FORTRAN
- Oracle Call Interface
- Oracle Precompiler and Oracle Call Interface Linking and Makefiles
- Static and Dynamic Linking with Oracle Libraries
- Using Signal Handlers
- XA Functionality

Overview of Oracle Precompilers

Oracle precompilers are application design tools used to combine SQL statements from 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 Oracle8, or any other ANSI SQL DBMS.

Table 5-1 lists which Oracle precompilers you can use with a given release of Oracle Server.

Table 5-1 Oracle Precompiler Release Compatibility

Oracle Server	Precompiler
Release 7.1	Release 1.6
Release 7.1.2	Release 1.7
Release 7.1.3	Release 2.0
Release 7.2.2	Release 2.1
Release 7.3.2	Release 2.2
Release 7.3.3	Release 2.2.3
Release 8.0.3	Release 8.0.3
Release 8.0.4	Release 8.0.4
Release 8.0.5	Release 8.0.5
Release 8.0.6	Release 8.0.6

Relinking Precompiler Executables

All precompiler executables are relinked using the `$ORACLE_HOME/precomp/lib/ins_precomp.mk` makefile. The `make` command uses the following convention:

```
$ make -f ins_precomp.mk relink EXENAME=executable
```

This command creates the new executable in the `$ORACLE_HOME/precomp/lib` directory, and then moves it to `$ORACLE_HOME/bin`. To create the new executable without moving it to `$ORACLE_HOME/bin`, use the following command:

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

Table 5–2 lists the executable names that you can specify for the `make` command.

Table 5–2 Products and Their Corresponding Executable Names

Product	Executable
Pro*C / C++ v8.0.6	`proc`
Pro*FORTRAN v1.8.28	`profor`
Object Type Translator v8.0.6	`ott`

For example, to relink the Pro*C / C++ executable, use the following command:

```
$ cd $ORACLE_HOME/precomp/lib
$ make -f ins_precomp.mk relink EXENAME=proc
```

Precompiler Configuration Files

There are three `.cfg` system configuration files in the `$ORACLE_HOME/precomp/admin` directory. Table 5–3 lists the `icfg` precompiler configuration files for each precompiler.

Table 5–3 System Configuration Files

Product	Configuration File
Pro*C / C++ v8.0.6	`pcscfg.cfg`
Pro*FORTRAN v1.8.28	`pccfor.cfg`
Object Type Translator v8.0.6	`ottcfg.cfg`

Issues Common to All Precompilers

The following issues are common to all precompilers.

Problems Linking Programs and Oracle Net8 Driver Libraries

If a sample program does not link properly because of unresolved symbols in `osnttt`, `osndnt`, or `osnasy`, either the Oracle Net8 drivers have not been installed or the Oracle Net8 libraries are missing from the makefile. To correct the problem, install the missing drivers or libraries, or put references to the Oracle Net8 driver libraries in the `proc.mk` makefile. The references must go in the link line after the `libnetwork.a` entry.

Uppercase to Lowercase Conversion

In languages other than C, the compiler converts an uppercase function or subprogram name to lowercase. This can cause "No such user exit" errors. In this case, verify that the function or subprogram name in your option file matches the case in the `iapxtb` table.

Vendor Debugger Programs

Precompilers and vendor-supplied debuggers might be incompatible. Oracle Corporation does not guarantee that a program run under a debugger will run the same way under an operating system.

Value of ireclen and oreclen

The `ireclen` and `oreclen` parameters do not have maximum values.

Supplementary Documentation

The following documents provide additional information about precompiler and interface features:

- *Programmer's Guide to the Pro*C/C++ Precompiler*
- *Pro*FORTRAN Supplement to Oracle Precompilers*
- *Programmer's Guide to the Oracle Call Interface*
- *Oracle8 Application Developer's Guide*

Pro*C/C++

For additional information regarding Pro*C/C++ release 8.0.6, see the $ORACLE_HOME/precomp/doc/proc2/readme.doc file.

Administering Pro*C/C++

The system configuration file for Pro*C/C++ is $ORACLE_HOME/precomp/admin/pcscfg.cfg.

> **See Also:** For more information, see the *Programmer's Guide to the Pro*C/C++ Compiler*.

Using Pro*C/C++

Before using Pro*C/C++, verify that the correct version of the operating system compiler is installed. The required version is documented in the *Oracle8 Installation Guide (32 Bit) for SGI IRIX*.

Demonstration Programs

Demonstration programs are provided to show the varied functionality of the Pro*C/C++ precompiler. There are three types of demonstration programs: C, C++, and Object programs - the latter demonstrate the new Oracle8 Object features.

All of the demonstration programs are located in $ORACLE_HOME/precomp/demo/proc directory, and all of them assume that the demonstration tables created by the $ORACLE_HOME/sqlplus/demo/demobld.sql script exist in the SCOTT schema with the password TIGER.

See "Enabling SQL*Plus Help" on page 4-5 for further information on building the demonstration programs using SQL*Plus. For further information on the demonstration programs see the *Programmer's Guide to the Pro*C/C++ Precompiler.*

Use the $ORACLE_HOME/precomp/demo/proc/demo_proc.mk makefile to create the demonstration programs. For example, to precompile, compile, and link the sample1 demonstration program, enter the following command.

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

Alternatively, use the following command which achieves exactly the same result, only with more explicit syntax.

```
$ make -f demo_proc.mk build OBJS=sample1.o EXE=sample1
```

By default, all programs are dynamically linked with the client shared library, $ORACLE_HOME/lib/libclntsh.so.

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

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

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

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

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

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

Some demonstration programs require you to run a SQL script from the $ORACLE_HOME/precomp/demo/sql directory. 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 calldemo demonstration program and run the required $ORACLE_HOME/precomp/demo/sql/calldemo.sql script, use the following command syntax:

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

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

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

You can also run the SQL scripts manually.

User Programs

You can run the $ORACLE_HOME/precomp/demo/proc/demo_proc.mk makefile to create user programs. The general syntax for linking a user program with demo_proc.mk is as follows:

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

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

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

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

- For C source, statically linked:

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

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

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

- For C++ source, statically linked:

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

> **Note:** If your program depends on libraries other than Oracle libraries, you might need to alter the demo_proc.mk makefile to include them.

Pro*FORTRAN

For additional information regarding Pro*FORTRAN 1.8.28, see the $ORACLE_HOME/precomp/doc/pro1x/readme.txt file.

Administering Pro*FORTRAN

The system configuration file for Pro*FORTRAN is $ORACLE_HOME/precomp/admin/pccfor.cfg.

Using Pro*FORTRAN

Before using Pro*FORTRAN, verify that the correct version of the operating system compiler is properly installed. The required version for your operating system is listed in the *Oracle8 Installation Guide (32 Bit) for SGI IRIX*.

Demonstration Programs

Demonstration programs are provided to show the varied functionality of the Pro*FORTRAN precompiler. All of the demonstration programs are located in the `$ORACLE_HOME/precomp/demo/profor` directory, and all of them assume that the demonstration tables created by the `$ORACLE_HOME/sqlplus/demo/demobld.sql` script exist in the SCOTT schema with the password TIGER.

See "Enabling the SQL*Plus Demonstrations" on page 4-3 for more information on building the demonstration programs using SQL*Plus. For more information on the demonstration programs see the *Pro*FORTRAN Supplement to Oracle Precompilers*.

Use the `$ORACLE_HOME/precomp/demo/profor/demo_profor.mk` makefile, to create the demonstration programs. For example, to precompile, compile, and link the `sample1` demonstration program, enter the following command:

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

Alternatively, use the following command, which achieves the same result, with more explicit syntax:

```
$ make -f demo_profor.mk build FORS=sample1.pfo EXE=sample1
```

By default, all programs are dynamically linked with the client shared library, `$ORACLE_HOME/lib/libclntsh.so`.

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

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

For some demonstration programs, you must run a SQL script located in the `$ORACLE_HOME/precomp/demo/sql` directory. To build a demonstration program and run the corresponding SQL script, the `make` macro argument `RUNSQL=run`, must be included on the command line. For example, to create the `sample11` demonstration program and run the required `$ORACLE_HOME/precomp/demo/sql/sample11.sql` script, use the following command syntax:

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

You can also run the SQL scripts manually.

User Programs

You can use the $ORACLE_HOME/precomp/demo/profor/demo_profor.mk makefile to create user programs. The general syntax for linking a user program with demo_profor.mk is as follows:

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

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

- For an executable that links dynamically with the client shared library:

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

- For a statically linked executable:

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

Oracle Call Interface

Before using the Oracle Call Interface (OCI), verify that the correct version of the compiler is properly installed. The required version for your operating system is specified in the*Oracle8 Installation Guide (32 Bit) for SGI IRIX*.

Demonstration Programs

Demonstration programs are provided that show varied functionality of the OCI. There are two types of demonstration programs: C and C++. All of the demonstration programs are located in the $ORACLE_HOME/rdbms/demo directory. Many of the demonstration programs assume that the demonstration tables created by the $ORACLE_HOME/sqlplus/demo/demobld.sql script exist in the SCOTT schema with the password TIGER.

See “Enabling the SQL*Plus Demonstrations” on page 4-3 for further information on building the demonstration programs using SQL*Plus. For further information on the demonstration programs see the *Programmer’s Guide to the Oracle Call Interface* and the program source for details of each program.

Use the $ORACLE_HOME/rdbms/demo/demo_rdbms.mk makefile to create the demonstration programs. For example, to compile and link the cdemo1 demonstration program, enter the following command:

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

Alternatively, you can use the following command which achieves the same result with more explicit syntax:

```
$ make -f demo_rdbms.mk build OBJS=cdemo1.o EXE=cdemo1
```

By default, all programs are dynamically linked with the client shared library, $ORACLE_HOME/lib/libclntsh.so.

To create all OCI C demonstration programs, enter the following command:

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

To create all OCI C++ demonstration programs, enter this command:

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

Some demonstration programs require you to run a SQL script manually before you execute the program. This script is located in the $ORACLE_HOME/rdbms/demo directory. In most cases, the SQL script name is the same as the program name with a .sql extension. For example, the SQL script for the program oci02 is oci02.sql.

Read the comments at the beginning of the program to determine the required SQL script, if any.

User Programs

You can use the $ORACLE_HOME/rdbms/demo/demo_rdbms.mk makefile to create user programs. The general syntax for linking a user program with demo_rdbms.mk is:

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

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

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

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

- For C source, statically linked:

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

To create the program `myprog` from the C++ source `myprog.cc`:

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

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

- For C++ source, statically linked:

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

Oracle Precompiler and Oracle Call Interface Linking and Makefiles

The following sections describe Oracle Precompiler and Oracle Interface Linking and makefiles.

Custom Makefiles

Oracle Corporation recommends that you use the `demo_`*`product`*`.mk` makefiles provided to link user programs as described in the specific product sections of this chapter. You must modify the provided makefile. If you choose to use a custom written makefile, note the following:

- Do not modify the order of the Oracle libraries.
  Oracle libraries are included on the link line more than once so that all symbols are resolved during linking.
- If you add your own library to the link line, add it to the beginning or to the end of the link line.
  Do not place user libraries between the Oracle libraries.
- If you choose to use a make utility such as `nmake` or GNU `make`, you should be aware of how macro and suffix processing differs from the make utility provided with SGI IRIX. Oracle makefiles have been tested and are supported with the SGI IRIX `make` utility.
- Oracle library names and the contents of those libraries are subject to change between releases. Always use the `demo_`*`product`*`.mk` makefile that ships with the current release as a guide to determine which libraries are necessary.

Undefined Symbols

Undefined symbol error messages similar to the following are commonly used when linking a program:

```
$ make -f demo_proc.mk sample1
Undefined                       first referenced
 symbol                             in file
sqlcex                              sample1.o
sqlglm                              sample1.o
ld: fatal: Symbol referencing errors. No output written to sample1
```

This error occurs when the linker cannot find a definition for a referenced symbol. Generally, the solution for this type of problem is to ensure that the library or object file containing the definition exists on the link line and that the linker is searching the correct directories for the file.

Oracle provides a utility called `symfind` to assist in locating a library or object file where a symbol is defined. The following output example shows `symfind` locating the symbol `sqlcex`:

```
$ symfind sqlcex

SymFind - Find Symbol <sqlcex> in <**>.a, .o, .so
------------------------------------------------------
Command:         /u01/app/oracle/product/8.0.6/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/8.0.6
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.so
[35]    |         0|      44|FUNC |GLOB |0    |5      |sqlcex
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ./lib/libsql.a
```

Static and Dynamic Linking with Oracle Libraries

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

With dynamic linking, the executing code partly resides in the executable program, and also resides in libraries that are linked by the application dynamically at runtime. Libraries that are linked at runtime are called dynamic or shared libraries. There are two primary benefits of dynamic linking; smaller disk requirements and smaller main memory requirements.

1. Smaller Disk Requirements

   Different applications, or different invocations of the same application, can use the same shared or dynamic library. As a result, the overall disk requirements are reduced.

2. Smaller Main Memory Requirements

   The same shared or dynamic library image (for example, the in-memory copy), can be shared by different applications. This means that a library needs to be loaded only once into the main memory and then multiple applications can use the same library. As a result, main memory requirements are reduced.

Oracle Shared Library

The Oracle shared library is `$ORACLE_HOME/lib/libclntsh.so`. If you use the Oracle provided `demo_`*`product`*`.mk` makefile to link an application, the Oracle shared library is used by default.

You might need to set the LD_LIBRARY_PATH environment variable so that the runtime loader can find the Oracle shared library at process start up. If you receive the following error when starting an executable, you must set LD_LIBRARY_PATH to the directory where the Oracle shared library exists:

```
% sample1
ld.so.1: sample1: fatal: libclntsh.so.1.0: can't open file: errno=2
Killed
```

Set LD_LIBRARY_PATH as follows:

```
% setenv LD_LIBRARY_PATH $ORACLE_HOME/lib
```

The Oracle shared library is created automatically during installation. If you need to recreate the Oracle shared library, exit *all* client applications using the Oracle shared library, including all Oracle client applications such as SQL*Plus and Recovery Manager, and run the following command logged in as the *`oracle`* user:

```
% cd $ORACLE_HOME/rdbms/lib
% make -f ins_rdbms.mk client_sharedlib
```

Using Signal Handlers

This section describes signals Oracle8 uses for two-task communication. It also explains how to set up your own signal handlers.

Signals

Signals are installed in a user process when you connect to the database, and are de-installed when you disconnect.

Table 5–4 describes the signals that Oracle8 uses for two-task communications:

Table 5–4 Signals for Two-Task Communications

SIGCONT	used by the pipe two-task driver to send out-of-band breaks from the user process to the `oracle` process.
SIGINT	used by all two-task drivers to detect user interrupt requests. SIGINT is not caught by `oracle`; it is caught by the user process.
SIGPIPE	used by the pipe driver to detect end-of-file on the communications channel. When writing to the pipe, if no reading process exists, a SIGPIPE signal is sent to the writing process. SIGPIPE is caught by both the `oracle` process and the user process.
SIGCLD	used by the pipe driver. SIGCLD is similar to SIGPIPE, but only applies to user processes, not `oracle` processes. When an `oracle` process dies, the UNIX kernel sends a SIGCLD to the user process (**wait**() is used in the signal handler to see if the server process died). SIGCLD is not caught by `oracle`; it is caught by the user process.
SIGTERM	used by the pipe driver to signal interrupts from the user side to the `oracle` process. This occurs when the user presses the interrupt key [Ctrl]+[c]. SIGTERM is not caught by the user process; it is caught by `oracle`.
SIGIO	used by Oracle Net8 protocol adapters to indicate incoming networking events.
SIGURG	used by the Oracle Net8 TCP/IP drivers to send out-of-band breaks from the user process to the `oracle` process.

The listed signals affect precompiler applications. You can install one signal handler for SIGCLD (or SIGCHLD) and SIGPIPE when connected to the `oracle` process. You can have multiple signal handlers for SIGINT as long as the osnsui() routine is called to set this up. You can install as many signal handlers as you want for other signals. If you are not connected to the *`oracle`* process**,** you can have multiple signal handlers.

Sample Signal Routine

The following example shows how you can set up your own signal routine and the catching routine. For SIGINT, use osnsui() and osncui() to register and delete signal-catching routines.

```
/* 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 can be cleared with osncui. Note that there might 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.
*/
```

The following is a template for using osnsui() and osncui() in an application program:

```
/*
** My own user interrupt handler.
*/
void sig_handler()
{
...
}

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

   int handle, err;
   ...

   /* set up my user interrupt handler */
   if (err = osnsui(&handle, sig_handler, (char *) 0))
```

```
{
/* if the return value is non-zero, an error has occurred
Do something appropriate here. */
...
}
...
/* clear my interrupt handler */
if (err = osncui(handle))
{
/* if the return value is non-zero, an error has occurred
Do something appropriate here. */
...
}
...
}
```

XA Functionality

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

Oracle8 does not support Oracle7 release 7.1.6 XA calls, although it does support release 7.3 XA calls, therefore TP-monitor XA applications using release 7.1.6 XA calls must be relinked with the Oracle8 XA library. The Oracle8 XA calls are defined in both the shared library `$ORACLE_HOME/lib/libclntsh.so` and the static library `$ORACLE_HOME/lib/libclient.a`.

6

Configuring Oracle Net8

This chapter describes how to configure Oracle Net8 after you have installed it. For information about installing Oracle Net8, see Chapter 1 and Chapter 3 of the *Oracle8 Installation Guide (32 Bit) for SGI IRIX*. This chapter contains the following sections:

- Supplementary Documentation
- Net8 Files and Utilities
- BEQ Protocol Adapter
- IPC Protocol Adapter
- RAW Protocol Adapter
- TCP/IP Protocol Adapter
- Oracle Enterprise Manager Intelligent Agent
- Oracle Advanced Networking Option
- Sample Network Configuration Files

Supplementary Documentation

The following documents fully describe Oracle Net8 features:

- *Oracle Net8 Administrator's Guide*
- *Oracle Networking Quick Reference Card for Net8*
- *Oracle Advanced Networking Option Administrator's Guide*
- *Oracle Security Server Guide*
- *Oracle Cryptographic Toolkit Programmer's Guide*

Supplementary Information in README Files

Table 6–1 shows the location of readme files for various bundled products. The readme files describe changes since the last release.

Table 6–1 Location of README Files for Oracle Products

Product	README File
Net8	`$ORACLE_HOME/network/doc/README.Net8`
Advanced Networking Option	`$ORACLE_HOME/network/doc/README.ANO`
Oracle Intelligent Agent	`$ORACLE_HOME/network/doc/README.oemagent`

Net8 Files and Utilities

This section describes the files and utilities that you can use to configure Oracle Net8 products.

Location of Net8 Configuration Files

The default directory for Net8 configuration files is `/var/opt/oracle` on SGI IRIX systems.

Oracle Net8 and Connection Manager search the following locations for global files in the following order:

1. The directory specified by the `$TNS_ADMIN` environment variable
2. The `/etc` or `/var/opt/oracle` directory
3. The `$ORACLE_HOME/network/admin` directory (to search for the `sqlnet.ora` file)

If your files are not in the default directory, use the TNS_ADMIN environment variable in the startup files of all network users to specify a different location.

For the Bourne and Korn shell, enter:

```
$ TNS_ADMIN=directory_name
$ export TNS_ADMIN
```

For the C shell, enter:

```
% setenv TNS_ADMIN directory_name
```

For each system level configuration file, there is also a local configuration file (stored in the $HOME directory). The settings in the local file override the settings in the system level file.

The local configuration file for `sqlnet.ora` is `$HOME/.sqlnet.ora`. The local configuration file for `tnsnames.ora` is `$HOME/.tnsnames.ora`. Syntax for these files is identical to that of the corresponding system files.

Sample Configuration Files

Examples of the `cman.ora`, `listner.ora`, `names.ora`, `sqlnet.ora`, and `tnsnames.ora` configuration files are located in the `$ORACLE_HOME/network/admin/samples` directory.

The Adapters Utility

Net8 provides support for various network protocols and naming methods. They are linked into particular executables and provide the interface between network protocols and Net8. To display installed Net8 protocol adapters, enter:

```
% adapters
```

To display protocol adapters linked with a specific executable, enter:

```
% adapters executable
```

For example, the following command displays the Net8 protocols linked with the `oracle` executable:

```
$ adapters oracle
Net8 Protocol Adapters linked with oracle are:
BEQ Protocol Adapter
IPC Protocol Adapter
TCP/IP Protocol Adapter
RAW Protocol Adapter
Net8 Naming Adapters linked with oracle are:
Oracle TNS Naming Adapter
Oracle Naming Adapter
Oracle Advanced Security/Networking Security products linked with oracle are:
```

Multi-Threaded Server

For information on the Multi-Threaded Server, see *Oracle8 Concepts* and the *Oracle8 Administrator's Guide.*

Net8 Assistant

The Net8 Assistant (`$ORACLE_HOME/bin/net8asst.sh`) requires Java 1.1.6. When you run the Net8 Assistant command script, the JRE command script is called explicitly. The JRE command script is located in the `/usr/java/bin/jre` directory.

For further information on Net8 Assistant, see the *Oracle Net8 Administrator's Guide.*

Oracle Net8 Protocol Adapters

The supported protocol adapters for Net8 version 8.0.6 on SGI IRIX are the BEQ protocol adapter, the IPC protocol Adapter, the RAW protocol adapter, and the TCP/IP protocol adapter.

Before you install the TCP/IP Net8 protocol adapter, you must install and configure the appropriate operating system software. Refer to the *Oracle8 Installation Guide (32 Bit) for SGI IRIX* for requirements. The BEQ and IPC Net8 protocol adapters do not have any specific operating system requirement.

The IPC and TCP/IP Net8 protocol adapters each have a protocol-specific ADDRESS specification that is used for Net8 configuration files and for the MTS_LISTENER_ADDRESS database initialization parameter (`init.ora`). See the ADDRESS specification heading under each protocol adapter section in this chapter for details.

Table 6–2 shows a summary of ADDRESS specifications for each protocol adapter.

Table 6–2 ADDRESS Specification Summary

Protocol Adapter	ADDRESS Specification
BEQ	(ADDRESS = (PROTOCOL = BEQ) (PROGRAM = ORACLE_HOME/bin/oracle) (ARGV0 = oracleORACLE_SID) (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))') (ENVS = 'ORACLE_HOME=*ORACLE_HOME*,ORACLE_SID=*ORACLE_SID*'))
IPC	(ADDRESS = (PROTOCOL=IPC) (KEY=*key*))
RAW	N / A
TCP / IP	(ADDRESS = (PROTOCOL=TCP) (HOST=*hostname*) (PORT=*port_id*))

BEQ Protocol Adapter

The BEQ protocol adapter is both a communications mechanism and a process-spawning mechanism. It requires that the client and server be on the same system. The BEQ protocol adapter is used unless a service name is specified by the user directly through the command line or login window, or indirectly through an environment variable such as TWO_TASK. The BEQ protocol adapter always uses a dedicated server. The Multi-Threaded Server is never used. This dedicated server is started automatically by the BEQ protocol adapter, which waits for the server process to start and attach to an existing SGA. If the startup of the server process is successful, the BEQ protocol adapter then provides inter-process communication through UNIX pipes.

An important feature of the BEQ protocol adapter is that no listener is required for its operation. The protocol adapter is linked into the client tools and directly starts its own server process without outside interaction. However, the BEQ protocol adapter can be used only when the client program and Oracle8 are installed on the

same computer. The BEQ protocol adapter is always installed and always linked to all client tools and to the Oracle8 server.

Specifying a BEQ ADDRESS

The BEQ protocol adapter connection parameters are part of the ADDRESS keyword-value pair. You can enter the parameters in any order.

```
(ADDRESS =
(PROTOCOL = BEQ)
(PROGRAM = ORACLE_HOME/bin/oracle)
(ARGV0 = oracleORACLE_SID)
(ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
(ENVS = 'ORACLE_HOME=ORACLE_HOME,ORACLE_SID=ORACLE_SID')
)
```

Table 6–3 describes the syntax for BEQ protocol adapter connection parameters.

Table 6–3 Syntax for BEQ Protocol Adapter Connection Parameters

Parameter	Description
PROTOCOL	The protocol adapter to use. The value is `beq`. It is not case sensitive.
PROGRAM	The full path to the `oracle` executable.
ARGV0	The name of the process as it appears in a `ps` listing. The recommended value is `oracle`*ORACLE_SID*.
ARGS	`'(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))'`
ENVS	Environment specification where *ORACLE_HOME* is the full path to the *ORACLE_HOME* directory of the database to connect to, and *ORACLE_SID* is the system identifier of the database to connect to.

Example 6–1 BEQ ADDRESS Specifying a Client

```
(ADDRESS =
(PROTOCOL = BEQ)
(PROGRAM = /u01/app/oracle/product/8.0.6/bin/oracle)
(ARGV0 = oracleV806)
(ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
(ENVS = 'ORACLE_HOME=/u01/app/oracle/product/8.0.6,ORACLE_SID=V806')
)
```

The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file.

IPC Protocol Adapter

The IPC protocol adapter is similar to the BEQ protocol adapter in that it can only be used when the client program and the Oracle8 Server are installed on the same system. The IPC protocol adapter differs from the BEQ protocol adapter in that it can be used with dedicated server and multi-threaded server configurations. The IPC protocol adapter requires a network listener for its operation. The IPC protocol adapter is always installed, and always linked in to all client tools and to Oracle8.

For the IPC protocol adapter, the location of the UNIX Domain Socket (IPC) file on UNIX systems changed after Oracle7 release 7.1. Therefore, if you have Oracle7 release 7.1 installed on the same system as Oracle8, and you attempt to make an IPC connection between the two instances, the connection might fail. The solution to this problem is to make a symbolic link between the directory where the IPC file used to be stored (`/var/tmp/o`) to where it is now stored (`/var/tmp/.oracle`).

Specifying an IPC ADDRESS

The IPC protocol adapter connection parameters are part of the ADDRESS keyword-value pair. You can enter the parameters in any order.

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

Table 6–4 describes the syntax for IPC protocol adapter connection parameters.

Table 6–4 Syntax for IPC Protocol Adapter Connection Parameters

Parameter	Description
PROTOCOL	Protocol adapter to use. The value is `ipc`. It is not case sensitive.
KEY	Service name of database or database identifier (ORACLE_SID).

Example 6–2 IPC ADDRESS Specifying a Client

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

The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file.

RAW Protocol Adapter

When data is transferred back and forth between a client and a server, Net8 adds its own header information onto every packet (a block of information sent over the network). Through the Raw Transport feature, Net8 can now minimize header information on each packet going over the network.

After the connection is established, two types of information flow over the network: data and break handling. The connection packets need the Net8 header information on them to establish the connection correctly. However, after the connection is established, all data packets are stripped of their Net8 header information and passed directly to the operating system, bypassing the Net8 and protocol layers of Net8. The performance of the connection is increased because of fewer protocol stack layers for the data to flow through and fewer bytes that are transmitted over the network.

This feature is transparently enabled when it is required. If no existing features require that header information be transmitted, the headers are stripped. For example, raw transport would not be enabled when you use encryption and authentication which require information to be sent with each packet of information.

This feature requires no configuration. Net8 determines if the conditions are met and then transparently switches to Raw Transport mode.

TCP/IP Protocol Adapter

Oracle Corporation recommends that you reserve a port for your listener in the `/etc/services` file of each Net8 node on the network. The default port is 1521. The entry lists the listener name and the port number; for example:

```
listener      1521/tcp
```

In this example *`listener`* is the name of the listener, as defined in the `listener.ora` file.

Reserve more than one port to start more than one listener.

Specifying a TCP/IP ADDRESS

The TCP/IP protocol adapter connection parameters are part of the ADDRESS keyword-value pair. You can enter the three parameters in any order. For example:

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

Table 6–5 describes the syntax for TCP/IP protocol adapter connection parameters.

Table 6–5 Syntax for TCP/IP Protocol Adapter Connection Parameters

Parameter	Description
PROTOCOL	The protocol adapter to use. The value is `tcp`. It is not case sensitive.
HOST	The host name or the host IP address.
PORT	The TCP/IP port. Either a number or the name specified in the `/etc/services` file. Oracle Corporation recommends a value of 1521.

Example 6–3 shows a sample TCP/IP ADDRESS:

Example 6–3 TCP/IP ADDRESS Specifying a Client

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

The last field could be specified by name, for example, (PORT=listener)**.** The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file.

Oracle Enterprise Manager Intelligent Agent

For information on Agent-Service Discovery and Auto-Configuration, see the *Oracle Enterprise Manager Configuration Guide*.

Debugging Tcl Scripts

The `oratclsh` executable is provided for debugging your Tcl scripts. Before executing `oratclsh`, set the TCL_LIBRARY environment variable to the `$ORACLE_HOME/network/agent/tcl` directory.

> **See Also:** See the *Oracle Enterprise Manager Application Developer's Guide* for more information.

Oracle Advanced Networking Option

The following sections provide information on configuring Oracle Advanced Networking Option.

.bak Files

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

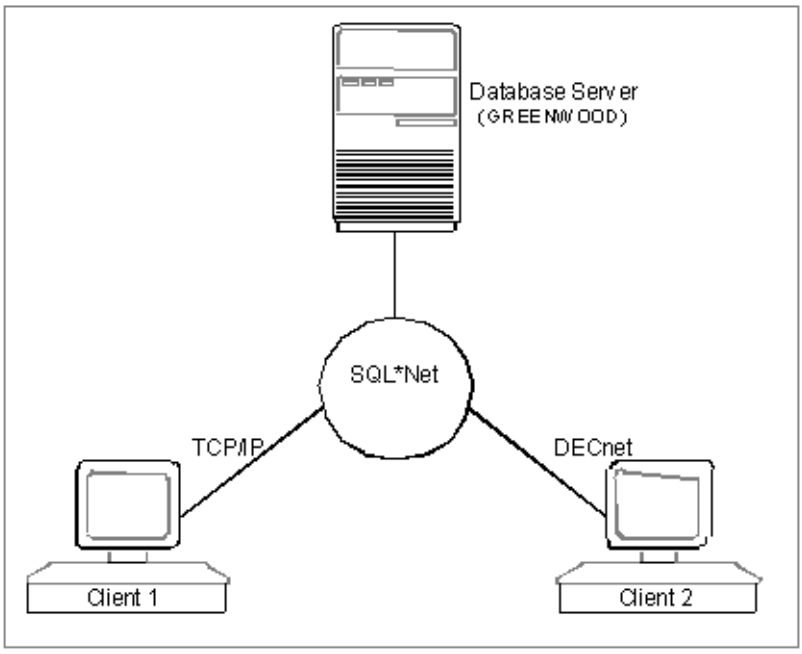
Security and Single Sign-On

For details on configuring Security and Single Sign-On, see the *Oracle Advanced Networking Option Administrator's Guide*.

Sample Network Configuration Files

Figure 6–1 shows the relationship between Oracle protocol adapters and the operating system protocol layer. The server has two Oracle instances linked to protocol adapters.

Figure 6–1 Sample Client/Server Network



The configuration files created automatically by Network Assistant are in three groups: one set for the server (hostname 'GREENWOOD'), and one set on each of the clients. The sample files generated are listed in the following sections.

Server-side Configuration Files

This section provides examples of the files that must be installed on the server system in an Oracle Net8 configuration.

Example 6-4 The listener.ora File

```
#################
# Filename......: listener.ora
# Name..........: GREENWOOD.world
# Date..........: 14-FEB-96 15:47:51
#################
LISTENER =
  (ADDRESS_LIST =
        (ADDRESS=
          (PROTOCOL=IPC)
          (KEY= GREEN.world)
        )
        (ADDRESS=
          (PROTOCOL=IPC)
          (KEY= GREEN)
        )
        (ADDRESS =
          (COMMUNITY = tcpcomm.world)
          (PROTOCOL = TCP)
          (Host = GREENWOOD)
          (Port = 1526)
        )
  )
STARTUP_WAIT_TIME_LISTENER = 0
CONNECT_TIMEOUT_LISTENER = 10
TRACE_LEVEL_LISTENER = OFF
SID_LIST_LISTENER =  (SID_LIST =
    (SID_DESC =
      (SID_NAME = GREEN)
      (ORACLE_HOME = /oracle/green)
      (PRESPAWN_MAX = 10)
    )
  )
```

Example 6-5 The sqlnet.ora File

```
#################
# Filename......: sqlnet.ora
# Name..........: GREENWOOD.world
# Date..........: 14-FEB-96 15:47:51
#################
AUTOMATIC_IPC = ON
TRACE_LEVEL_CLIENT = OFF
SQLNET.EXPIRE_TIME = 0
NAMES.DEFAULT_DOMAIN = world
NAME.DEFAULT_ZONE = world
SQLNET.CRYPTO_SEED = "-864046921-864011456"
SQLNET.AUTHENTICATION_SERVICES = (ALL)
```

Example 6-6 The tnsnames.ora File

```
#################
# Filename......: tnsnames.ora
# Name..........: LOCAL_REGION.world
# Date..........: 14-FEB-96 15:47:51
#################
G_TCP.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (COMMUNITY = tcpcomm.world)
          (PROTOCOL = TCP)
          (Host = GREENWOOD)
          (Port = 1526)
        )
    )
    (CONNECT_DATA =
       (SID = GREEN)
       (GLOBAL_NAME = G_TCP.world)
    )
  )
```

TCP/IP Client Configuration Files

This section provides examples of the files that must be installed on the client system in an Oracle Net8 configuration.

Example 6-7 The sqlnet.ora File

```
#################
# Filename......: sqlnet.ora
# Name..........: tcpcomm.world
# Date..........: 14-FEB-96 15:47:51
#################
AUTOMATIC_IPC = ON
TRACE_LEVEL_CLIENT = OFF
SQLNET.EXPIRE_TIME = 0
NAMES.DEFAULT_DOMAIN = world
NAME.DEFAULT_ZONE = world
SQLNET.CRYPTO_SEED = "-864046921-864011456"
SQLNET.AUTHENTICATION_SERVICES = (ALL)
```

Example 6-8 The tnsnames.ora File

```
#################
# Filename......: tnsnames.ora
# Name..........: LOCAL_REGION.world
# Date..........: 14-FEB-96 15:47:51
#################
GREEN.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (COMMUNITY = tcpcomm.world)
          (PROTOCOL = TCP)
          (Host = GREENWOOD)
          (Port = 1526)
        )
    )
    (CONNECT_DATA =
       (SID = GREEN)
       (GLOBAL_NAME = GREEN.world)
    )
  )
```

Configuring the Oracle TCP/IP Protocol Adapter

Transmission Control Protocol / Internet Protocol (TCP / IP) is a family of related protocols. TCP / IP provides the reliable data-transfer services from one point to another and IP dispatches information around a network.

The TCP / IP protocol views the network as a two-way data transmission medium. The network provides connection-oriented interprocess communication between pairs of processes in host computers attached to interconnected computer networks.

The application or client process initiates the TCP / IP connection with a remote host process by specifying a host IP address and a TCP port (or entry point) on the host. When they are connected, the pair of communicating processes send and receive data in a continuous byte stream.

Configuring the Listener

For each node in your network, Oracle Corporation recommends that you reserve a port for your Oracle Net8 listener in the `/etc/services` file. This file defines the Oracle Net8 listener port. Typically, this is port 1521. The entry should list the listener name and the port number, for example:

```
listener      1521/tcp
```

In this example, *`listener`* is the name of the listener, as defined in the `listener.ora` file.

Reserve more than one port to start more than one listener.

Specifying the TCP/IP Protocol Adapter Address

The Oracle TCP / IP protocol adapter allows TNS and its applications to integrate with the TCP / IP communications protocol. The TCP / IP protocol adapter implements a standard interface used to resolve the equivalent communication functions of the TCP / IP protocol and TNS.

After the TCP / IP protocol and Oracle TCP / IP protocol adapter are installed on your system, you can use TCP / IP parameters with the TNS connect descriptors to identify nodes in a TCP / IP community.

The TCP/IP protocol adapter connection parameters are part of the ADDRESS keyword-value pair. You can enter the name of the TCP/IP community and the three TCP/IP parameters in any order:

```
(ADDRESS=
[(COMMUNITY=community_name)]
(PROTOCOL=TCP)
(HOST=hostname)
(PORT=port_id)
```

Table 6–6 describes the syntax for the BEQ protocol connection parameters.

Table 6–6 Syntax for BEQ Protocol Connection Parameters

Parameter	Description
PROTOCOL	The protocol adapter to be used. The value is BEQ. It is not case sensitive.
PROGRAM	The full path to the `oracle` executable.
ARGV0	The name of the process as it appears in a `ps` listing. The recommended value is oracle*ORACLE_SID*.
ARGS	`'(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))'`
ENVS	Environment specification where *ORACLE_HOME* is the full path to the *ORACLE_HOME* directory of the database to connect, and *ORACLE_SID* is the system identifier of the database to connect.

Example 6-9 shows a sample TCP/IP ADDRESS:

Example 6-9 Specifying a Client in the TCP/IP Address

```
(ADDRESS=
    (COMMUNITY=TCP.MFG.ACME)
    (PROTOCOL=TCP)
    (HOST=MADRID)
    (PORT=1521))
```

The port can be specified by name, for example, PORT=listener**.** The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file.

7

Running Oracle Data Cartridge Demonstrations on SGI IRIX

This chapter provides information on issues specific to SGI IRIX systems running the Oracle Data Cartridges that are supplied with this release of Oracle8. It contains the following sections:

- Issues Common to Data Cartridges
- Oracle8 Time Series Cartridge
- Oracle8 Visual Information Retrieval Cartridge
- Oracle8 Image Cartridge

Issues Common to Data Cartridges

Data cartridges require a C compiler to build the sample applications after installation.

> **See Also:** For more information about Oracle data cartridges, refer to the *Oracle Enterprise Manager Configuration Guide* and the documentation for the individual cartridge.

Oracle8 Time Series Cartridge

For more information on the Oracle8 Time Series Cartridge, refer to the *Oracle Time Series Cartridge User's Guide.*

Installing Time Series Cartridge Demonstrations

Time Series Cartridge demonstration files are stored in the `$ORACLE_HOME/ord/ts/demo` directory. This directory contains several demonstrations, each in its own subdirectory:

- The `usage` demonstration shows how to build a time-series object and call-time series functions. Run this demonstration first to create sample time series schemas used by the other demonstrations.
- The `extend` demonstration shows how to write and add customized time series functions.
- The `proc` demonstration shows how to access data stored with the cartridge using Pro*C.
- The `oci` demonstration shows how to access data via the Oracle Call Interface.
- The `dev2k` demonstration includes an Oracle Developer form which retrieves data using the cartridge.

Refer to the README files in the demonstration directories for more information about each demonstration.

Oracle8 Visual Information Retrieval Cartridge

For more information on the Oracle8 Visual Information Retrieval Cartridge, refer to the *Oracle Visual Information Retrieval Cartridge User's Guide.*

Creating the Demonstration

Follow the instructions in this section to create the demonstration programs for Oracle8 Visual Retrieval Cartridge using Server Manager.

1. Create the VIR demonstration directory:

```
% svrmgrl
SVRMGRL> CONNECT INTERNAL;
SVRMGRL> CREATE OR REPLACE DIRECTORY virdemodir
    AS '$ORACLE_HOME/ord/vir/demo';
```

2. Grant privileges on the directory to PUBLIC:

```
SVRMGRL> GRANT READ ON DIRECTORY virdemodir TO PUBLIC
    WITH GRANT OPTION;
SVRMGRL> EXIT;
```

3. Make the `virdemo` program:

```
% cd $ORACLE_HOME/ord/vir/demo
% make -f demo_ordvir.mk virdemo
```

Oracle8 Image Cartridge

For more information on Oracle Image Cartridge, refer to the *Oracle Image Cartridge User's Guide.*

Creating the Demonstration

Follow the instructions in this section to create the Image Cartridge demonstration directory.

1. Create the Image Cartridge demonstration directory:

```
% svrmgrl
SVRMGRL> CONNECT INTERNAL;
SVRMGRL> CREATE OR REPLACE DIRECTORY imgdemodir
    AS '$ORACLE_HOME/ord/img/demo';
```

2. Grant privileges on the directory to PUBLIC:

```
SVRMGRL> GRANT READ ON DIRECTORY imgdemodir TO PUBLIC
    WITH GRANT OPTION;
SVRMGRL> EXIT;
```

3. Make the `imgdemo` program:

```
% cd $ORACLE_HOME/ord/img/demo
% make -f demo_ordimg.mk imgdemo
```

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X