Oracle9*i*

Administrator's Reference

Release 2 (9.2.0.2) for hp OpenVMS Alpha

Part No. B10506-01

December 2002

ORACLE®

Oracle9*i* Administrator's Reference, Release 2 (9.2.0.2) for hp OpenVMS Alpha

Part No. B10506-01

Primary Author: Platform Technologies Division

Contributing Authors: Pierre Krabbendam, Tom Montgomery, Dave Hayter, Sam Ravindhran, Kannan Viswanathan, David Robles, John DeWitt

Contents

8 Using Oracle Precompilers and the Oracle Call Interface

C HP OpenVMS Alpha Process Control

D Running Oracle Text and Oracle Spatial Demonstrations

Index

Send Us Your Comments

Oracle9*i* Administrator's Reference, Release 2 (9.2.0.2) for hp OpenVMS Alpha

Part No. B10506-01

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 any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us at the following e-mail address:

osdwrite_us@oracle.com

If you would like a reply, please provide your name, address, and telephone number.

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

Preface

Purpose

This guide and the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha* provide instructions for installing and configuring Oracle9*i* Release 2 (9.2.0.2) on OpenVMS systems. Documentation for products included with the software is in the Oracle9*i* Generic Documentation Set.

Audience

This document is intended for anyone responsible for administering Oracle9*i* Release 2 (9.2.0.2) on OpenVMS systems.

Oracle9*i* and Oracle9*i* Enterprise Edition

Unless noted otherwise, features and functionality described in this document are common to both Oracle9*i* and Oracle9*i* Enterprise Edition.

Typographic Conventions

All output is shown as it actually appears. For input, refer to the list of conventions and their meanings in the following table.

`monospace`	`Monospace` type indicates OpenVMS DCL commands, directory names, usernames, pathnames, and filenames.
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 logical, including logical portions of filenames. It is also used for emphasis.
UPPERCASE	Uppercase letters indicate Structured Query Language (SQL) reserved words, initialization parameters, and environment logicals.

Command Syntax

OpenVMS command syntax appears in `monospace` font. The "`$`" character at the beginning of OpenVMS command examples need not be entered at the prompt.

hyphen -	A hyphen indicates a command that is too long to fit on a single line. Enter the line as printed or enter it as a single line without a hyphen:
	`copy disk$server25:[oracle902.rdbms.admin]*.sql-`
	`disk$server2:[oracle.rdbms.admin]`

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 logical: *library_name*

vertical line | A vertical line indicates a choice within braces or brackets:

```
SIZE filesize [K/M]
```

Accessing Installed Documentation

Oracle9*i* for hp OpenVMS Alpha Documentation

Oracle9*i* for hp OpenVMS Alpha documentation includes this guide and the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha*.

To access the documentation in HTML and PDF formats, use a browser to open the `products.htm` file at the top level of the Oracle9*i* CD-ROM. This file contains links to product and OpenVMS-specific documentation.

Oracle Product Documentation

Oracle9*i* product documentation is on the Oracle9*i* Generic Documentation CD-ROM. Instructions for accessing and installing the documents on the CD-ROM are found in the README file on the top level directory of the CD-ROM.

Related Documentation

If you are unfamiliar with the concepts or terminology associated with relational database management systems, then refer to *Oracle9i Concepts* before beginning your installation. Read the *Quick Installation Procedure* for an overview of the installation process. Use the *Installation Checklist* to ensure that you have required information and that you have completed necessary pre-installation steps for a successful installation.

Information about system administration and tuning for a production database system is provided in these documents:

- *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha*
- *Oracle9i Database Administrator's Guide*
- *Oracle Net Services Guide*
- *Oracle9i Database Performance Guide and Reference*

Information about migrating from a previous version of the Oracle Server is provided in *Oracle9i Migration*.

Oracle Services and Support

Oracle's corporate web page is at the following address:

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

Oracle Corporation offers a wide range of services to help facilitate corporate system solutions, including Oracle Education courses, Oracle Consulting services, and Oracle Support Services from the website. In addition, Oracle Corporation provides free trial software, updates on Oracle products and service, and technical brochures and data sheets.

Oracle Support Services

Technical Support registration and contact information worldwide is available at the following address:

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

At Oracle's support site, you will find templates to help you prepare information about your problem before you call so that you may be helped more quickly. You will also need your CSI number (if applicable) or complete contact details, including any special project information.

Oracle*MetaLink*

Oracle Oracle*MetaLink* is Oracle Corporation's web service for technical information. Members of Oracle*MetaLink* can search for updates, alerts, patches, and other information about products, releases, and operating systems, or set preferences to be notified automatically of new information. Oracle*MetaLink* offers a variety of services to assist in setting up and administrating Oracle products, including procedures, scripts, commentary, and tuning and configuration best-practices bulletins. Please logon to Oracle*MetaLink* before installing or administrating your product to search for up to date information about Oracle9*i* Release 2 (9.2.0.2) for hp OpenVMS Alpha.

In addition, Oracle*MetaLink* offers forums for information sharing among Oracle customers, and direct communication with Oracle Support Services. Oracle*MetaLink* is available to Product Support Customers at no extra cost. Sign up for free membership for this service at the following site:

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

Use your Support Access Code (SAC) number to register.

Oracle Products and Other Documentation

For U.S.A. customers, the Oracle Store is at:

`http://store.oracle.com`

Links to Stores in other countries are provided from this site.

Oracle Technology Network

Register with the Oracle Technology Network (OTN) at:

`http://otn.oracle.com`

OTN delivers technical papers, code samples, product documentation, self-service developer support, and Oracle's key developer products to enable rapid development and deployment of applications built on Oracle technology.

Product documentation can be found at:

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

Customer Service

Oracle Support Service contacts are listed at:

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

Education and Training

Training information and worldwide schedules are available from:

`http://education.oracle.com`

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Support for Hearing and Speech Impaired Customers

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

- For technical questions, call:

  1.800.446.2398

- For non-technical questions, call:

  1.800.464.2330

Part I

Administering Oracle9*i* on OpenVMS

Part 1 of this guide consists of the following chapters:

1

Administering Oracle9*i* on HP OpenVMS Alpha

This chapter provides information on administering Oracle9*i* on OpenVMS. It contains the following sections:

1.1 Overview

You must set Oracle9*i* environment variables (logicals), parameters, and user settings (UAF), as well as O/S settings (SYSGEN) for Oracle9*i* to work. This chapter describes the various settings for Oracle9*i* on HP OpenVMS Alpha.

In order for Oracle9*i* to work correctly, you must define VMS logicals to set up the the Oracle9*i* environment; you must also set the appropriate parameters and user settings.

1.1.1 Oracle9*i* Code

Oracle9*i* code consists of several object libraries that are used to form the Oracle9*i* image during installation.

The code also consists of a shared Oracle client image linked during installation. For more information about shared Oracle client image, refer to the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha*.

Code for the Oracle9*i* is built to use 64-bit pointers to support very large System Global Areas. The code for clients, however, is built to use 32-bit pointers to maintain compatibility with existing client code. There are, therefore, both 32-bit and 64-bit versions of the object and shareable libraries installed. Oracle only supports 32-bit clients. Client applications may not be built with 64-bit pointers.

1.1.1.1 ORACLE.EXE File

When the Oracle9*i* Enterprise Edition is linked and installed, these routines reside as shareable code in OpenVMS global memory.

Occasionally, you will need to relink the image; for example, when new code is distributed or when a new release of HP OpenVMS Alpha is installed.

1.1.2 Oracle9*i* Instances

An Oracle9*i* instance is a combination of Oracle Server processes and memory buffers.

Because many instances can exist on one system or in one OpenVMS Cluster, you must assign every instance a unique one-to-six character system ID (SID). During the installation procedure, you create an instance when you create the initial database.

The SID that you assign to this instance becomes the default value of ORACLE_SID. The SID must be assigned to the logical name ORACLE_SID before the instance starts. To do this, run the following:

```
$ @orauser <SID>
```

1.1.2.1 System Global Area

All ORACLE operations use data stored in an area of shared memory called the System Global Area, commonly known as the SGA, that is allocated to each ORACLE instance. The size of the SGA is determined by the INIT.ORA file start-up parameters. After you create an instance, you can change the size of its SGA by shutting down the instance with the SQL*Plus® utility and modifying the values set in the INIT.ORA file as needed.

The size of the SGA is based on the values of the variable INIT.ORA parameters. These parameters determine:

- Allocation of ORACLE resources used by the processes that share the SGA
- Amount of data that might be maintained in the SGA

Consequently, parameter settings also determine the memory space needed to support these requirements. Increasing the value of these parameters can improve performance, but performance might also decrease if the SGA is so large that it consumes enough of the system memory that the system is forced to page portions of processes in and out of memory.

Oracle9*i* Release 2 (9.2.0.2) includes support for the Very Large Memory (VLM) 64-bit feature. This allows a large SGA that is limited only by the amount of physical memory available.

Data retrieved or inserted by user transactions is temporarily buffered in the SGA. Because this data resides in an area of memory accessible to all ORACLE processes, disk I/O is reduced and transaction time is significantly improved. The most significant structures in the SGA are the shared pool, database buffer pool, and redo log buffer.

1.1.2.1.1 shared pool

The shared pool contains shared cursors, stored procedures, SQL, PL/SQL blocks, and trigger code. The size of the shared pool is specified by the initialization parameter SHARED_POOL_SIZE. Larger values of this parameter improve performance in multi-user systems. Smaller values use less memory.

The limit for this parameter is determined by the size of your SGA. The shared pool must be at least 15 MB.

1.1.2.1.2 database buffer pool

Blocks of data retrieved by user transactions are read from the database file and then cached in the database buffer pool in the SGA. This data remains in the buffer pool (even after changes are committed) until more buffers are required; then, if the data has been modified, it is written to the database file(s).

The number of blocks that can be maintained in the buffer pool is determined by the initialization parameters DB_CACHE_SIZE or DB_nnK_CACHE_SIZE. For more information, refer to the *Oracle9i Database Administrator's Reference*.

1.1.2.1.3 redo log buffer

When data is modified, a record of the change (known as a redo entry) is generated in the redo log buffer. When changes to the data are committed, the redo entries in the buffer are written to the current redo log file. Redo log files provide for data recovery if media or system failure occurs before modified data is written from the database buffer to the database file.

The number of bytes that can be maintained in the redo log buffer is determined by the initialization parameter LOG_BUFFER.

1.1.2.2 Storing Data

Data is stored in database files. Each database must have at least one database file. Whenever you create a database, an initial database file is also created for the database.

During the installation procedure, you create one database file, typically in the `ORA_ROOT:[`<home>`.oradata.`<***dbname***>`]` directory, where <*dbname*> is the name you assign to the database. You can specify any directory for the system data file, and this directory does not necessarily need to be under ORACLE_HOME.

This initial file contains the data dictionary tables and all data entered by Oracle users (until you expand your database by creating tablespaces and adding data files).

Oracle Corporation recommends that the cluster size on the disk drive that will contain the database files be an integer multiple of the Oracle9*i* Enterprise Edition block size. For example, if the blocks are 2 KB, then the cluster size should be 2 KB, 8 KB, 12 KB, etc. Keep in mind, though, that cluster sizes are specified in terms of disk blocks (where one block = 512 bytes). Thus, a 2 KB cluster is an 4-block cluster.

A disk cluster size is the minimum unit of disk allocation. You determine the size when you initialize a disk.

1.1.2.3 Storing Changes to the Database

Changes made to the database are logged in the database buffer pool and in a file called a redo log. The changes recorded in the redo log provide for data recovery if media, software, or system failure occurs before the database buffers are written to the database files. Every database must have at least two redo log files so that another redo log will be available when the current log is filled.

Modified data is written from the database buffer pool to the database files when the current redo log fills or when the number of blocks in the redo log equals the value set by the INIT.ORA parameters LOG_CHECKPOINT_INTERVAL. Any event that causes the database buffers to be written is known as a checkpoint. The default value of the LOG_CHECKPOINT_INTERVAL parameter is 10,000 disk blocks (5MB). Setting the value is up to you; however, setting the LOG_CHECKPOINT_INTERVAL parameter initialization parameter to zero (in multiples of physical block size) eliminates interval checkpoints. This will reduce the checkpoint frequency and optimize runtime performance.

1.1.2.4 Using Redo Log Files

You can specify one of two modes for writing redo log files: ARCHIVELOG and NOARCHIVELOG. Using the redo logs in ARCHIVELOG mode allows data recovery in the event of media, software, and system failure.

Caution: If you are using NOARCHIVELOG mode when a media failure occurs, you will not be able to perform media recovery. You must use ARCHIVELOG mode in order to recover from media failure.

When a redo log file fills, the DBA must archive the log file to an offline file before the redo log file can be reused. (If it is not archived by the time all other redo log files are filled, then ORACLE operations are suspended until archiving is completed.) The DBA can archive the redo logs either manually or automatically.

In NOARCHIVELOG mode, data in the log file is overwritten when a redo log file must be reused. However, data is never overwritten until data in the database buffer has been written to the database file. Using the redo log files in NOARCHIVELOG mode ensures data recovery for software and system failure only.

During the Oracle Server installation procedure, you will create two redo log files named ORA_LOG1.RDO and ORA_LOG2.RDO. By default, these go into the ORA_DB directory, but you can choose an alternate directory. These log files are used in NOARCHIVELOG mode by default. You can change the mode to ARCHIVELOG. These files are also 100MB each by default; you can alter this size and specify different file names during the installation procedure if you want.

1.1.2.5 Using Logical Names

You can use logical names to specify the names of the database, redo log, and control files. Oracle Corporation recommends that you use system level logical names to name the devices where the database and redo log files reside, and that you specify full directory and filename paths for these files.

Control files store logical filenames as their translated equivalents, but do not translate concealed logical names.

Caution: Never use process-level concealed logical names to name any ORACLE database, redo log, or control file.

You can rename these files by using the ALTER DATABASE and ALTER TABLESPACE commands.

Note: Be careful if you plan to rename the files. Be sure you have sufficient knowledge of the ALTER DATABASE and ALTER TABLESPACE commands as discussed in the *Oracle9i Database Administrator's Reference*.

1.1.3 Logical Names Datafiles: Locations and Identifying by

Oracle datafiles may be placed in any location on any disk subject to the following restrictions:

- The datafiles or the directory that contains the datafiles cannot be owned by anyone with a group equal to or less than MAXSYSGROUP.
- The Oracle9*i* account must have write access to the location of the datafiles.
- Datafiles cannot be put in the root level directory of a disk.

If the directory is not specified, the default location for created datafiles is ORA_DB.

You can identify your datafiles by logical names rather than fully qualified filenames in your CREATE DATABASE or ALTER TABLESPACE statements. However, these logical names must be defined at the GROUP level or above, preferably at the SYSTEM level. Logical names at the PROCESS or JOB level **CANNOT** be used to identify datafiles. If you identify your datafiles by logical names, make sure these logical names are defined during system startup before you restart your databases after a reboot.

1.2 Relinking Executables

You can relink your product executables using the script that was created by Oracle Universal Installer. Relinking is necessary after applying any operating system patches, Oracle software patches or after an operating system upgrade.

The relink script manually relinks Oracle product executables, depending on the products that have been installed in the ORACLE_HOME directory.

1.3 Database Limits

Table 1–1 lists the default and maximum values for parameters in a `CREATE DATABASE` or `CREATE CONTROLFILE` statement.

Note: Interdependencies among these parameters may affect allowable values.

Table 1–1 CREATE CONTROLFILE and CREATE DATABASE Parameters

Parameter	Default	Maximum Value
MAXLOGFILES	32	254
MAXLOGMEMBERS	2	5
MAXLOGHISTORY	1600	5000
MAXDATAFILES	32	(999999999)
MAXINSTANCES	16	63

Table 1–2 lists the Oracle9*i* file size limits specific to OpenVMS.

Table 1–2 File Size Limits

File Type	Maximum Size
Data file	2^^64 Bytes (Single data file size)
Import/Export file	2,147,483,647
SQL*Loader file	2,147,483,647

1.4 Operating System Accounts and Groups

Special operating system accounts and groups are required by Oracle9*i*. Table 1–3 describes the "Oracle" and "SYSTEM" operating system accounts.

Table 1–3 OpenVMS Accounts

Name	Description
`oracle`	This account, called the Oracle software owner, is the operating system account that owns the Oracle9*i* software. The remainder of this document will refer to this account simply as the "Oracle user."
`SYSTEM`	The `system user` is a special operating system account with maximum privileges. This account is used to configure and install networking software, and create user accounts. This "user" is not to be confused with Oracle account SYSTEM.

1.5 Security

Oracle9*i* uses several features of the OpenVMS 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 Oracle9*i* improves security by dividing work (and address space) between the user program and the `oracle` server process. All database access is achieved using the shadow process and special authorizations in the `oracle` server process.

> **See Also:** For more information on security issues, refer to the *Oracle9i Administrator's Guide*.

1.5.1 Security for SQL*Plus Command "CONNECT AS SYSDBA"

Only the Oracle software owner and database administrator should have system privileges and the requirements for STARTUP, SHUTDOWN, and CONNECT AS SYSDBA.

1.5.2 Security for Database Files

The Oracle user should own the database files. Set the authorizations on these files to read/write by owner, and read-only for group or other users.

The Oracle user should own the directories containing the database files. For added security, revoke read permission from group and other users.

1.5.3 Customizing the Initialization File

The default initialization file (`init.ora`) is provided with the Oracle9*i* software. All Oracle9*i* instances assume these values if you do not 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.

Use the SQL*Plus command SHOW PARAMETERS to display the current values of these parameters on the system.

> **See Also:** The *Oracle9i Database Reference*, *Oracle9i Database Administrator's Guide* and *Oracle9i Tuning*.

Caution: The following INIT.ORA parameters are not applicable on OpenVMS. Do not change their default values.

- backup_tape_io_slaves
- dbwr_io_slaves
- disk_asynch_io
- tape_asynch_io
- db_writer_processes

Table 1–4 lists default initialization parameter values on OpenVMS and lists their default values and range of values.

Table 1–4 Initialization Parameters

Parameters	Default Value	Range of Values
BITMAP_MERGE_AREA_SIZE	1048576	65536 to unlimited
COMMIT_POINT_STRENGTH	1	0 to 255
CONTROL_FILES	none	none
CREATE_BITMAP_AREA_SIZE	8388608	65536 to unlimited
DB_nnK_CACHE_SIZE	100 MB	2 K to 32 K
DB_BLOCK_SIZE	2048	2 K to 32 K
DB_CACHE_SIZE	32MB	8 MB to unlimited
DB_FILES	200	1 to 2000000
DB_FILE_MULTIBLOCK_READ_COUNT	16	1 to the lower of the following: ▪ The value of the DB_nnK_CACHE_SIZE parameter divided by 4 ▪ 1048576 divided by the value of the DB_BLOCK_SIZE parameter
DISTRIBUTED_TRANSACTIONS	1/4 TRANSACTIONS	0 to unlimited
HASH_AREA_SIZE	2*SORT_AREA_SIZE	0 to unlimited
JAVA_POOL_SIZE	112MB	Between 1000000 and 1000000000
LOG_BUFFER	512 KB or 128 KB multiplied by the value of the CPU_COUNT parameter, which ever is higher	66560 to unlimited
LOG_CHECKPOINT_INTERVAL	0	0 to unlimited
DISPATCHERS	5	Between MAX_DISPATCHERS and PROCESSES

Table 1–4 (Cont.) Initialization Parameters

Parameters	Default Value	Range of Values
MAX_SHARED_SERVERS	2 multiplied by the value of the SHARED_SERVER parameter, if the value of the SHARED_SERVERS parameter is greater than 20, otherwise 20	Between SHARED_SERVERS and PROCESSES
SHARED_SERVERS	1, if DISPATCHERS is specified, else 0	Between 1 and PROCESSES
NLS_LANGUAGE	AMERICAN	Valid language names
NLS_TERRITORY	AMERICA	Valid territory names
OBJECT_CACHE_MAX_SIZE_PERCENT	10	0 to unlimited
OBJECT_CACHE_OPTIMAL_SIZE	100 KB	10 KB to unlimited
OPEN_CURSORS	300	1 to unlimited
OS_AUTHENT_PREFIX	ops$	Arbitrary string

The following two parameters have been desupported:

- DB_FILE_DIRECT_IO_COUNT
- HASH_MULTIBLOCK_IO_COUNT

Refer to the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha* for a complete list of desupported initialization parameters.

1.5.4 Running the orapwd Utility

You can use a password file to identify users that can use the SYSDBA and SYSOPER privileges when connecting to the database. To create the password file:

1. Log in as the Oracle user.
2. Use the `ORACLE_HOME:[BIN]orawpd` utility, which has the following syntax:

```
$ orapwd file=filename password=password entries=max_users
```

Table 1–5 describes the ***filename***, ***password*** and ***max_users*** syntax for executing `orapwd`:

Table 1–5 Syntax for Executing orapwd

Variable	Description
filename	Name of the file where password information is written. The name of the file must be `orapw`***sid***, and you must supply the full pathname. Its contents are encrypted and not user-readable. This parameter is mandatory. The password file is typically created in the `ora_db` directory.
password	This parameter sets the password for the SYS user. If you use an ALTER USER statement to change the password for the SYS user after you connect to the database, both the password stored in the data dictionary and the password stored in the password file are updated. This parameter is mandatory.
max_users	Maximum number of entries that you require the password file to accept.

See Also: the *Oracle9i Database Administrator's Guide* for more information on using the `orapwd` utility.

1.5.5 Password Management

For security reasons, the Database Configuration Assistant locks most Oracle user accounts after it creates the database. It does not lock the SYS, SYSTEM, or SCOTT accounts. You must unlock these accounts and change their passwords before logging into them. Use SQL*Plus to connect to the database as SYSDBA and enter the following command:

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

1.6 PL/SQL Gateway

The mod_plsql module is a PL/SQL gateway running within an Apache module in the middle tier server. It executes PL/SQL procedures in a backend Oracle server using OCI. The mod_plsql module currently supports only stateless PL/SQL web applications.

See Also: For information on developing web applications using PL/SQL, refer to *Oracle9i Application Server mod_plsql User's Guide*, which is the generic PL/SQL gateway documentation.

1.6.1 Installing the PL/SQL Gateway

The PL/SQL Gateway is installed as part of the Apache Install on OpenVMS. Please also refer to the `README_MODPLSQL.TXT` for more information on installing and configuring the PL/SQL gateway.

1.7 Oracle HTTP Server powered by Apache

Refer to Appendix D, "Installation and Configuration of Apache Server for OpenVMS" in the document, *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha* for complete information on installing and configuring Oracle HTTP Server powered by Apache.

1.8 Demonstration Files

This section describes how to build and run the SQL*Loader and PL/SQL demonstration programs installed with Oracle9*i*.

1.8.1 SQL*Loader Demonstrations

The following SQL*Loader demonstration files are included with Oracle9*i* in the `ora_rdbms_demo` directory. Run the demonstrations in numerical order:

ulcase1
ulcase2
ulcase3
ulcase4
ulcase5
ulcase6
ulcase7

To Create and Run a Demonstration

Run demonstrations while logged in as the user SCOTT/TIGER. Ensure that:

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

In the following steps, *n* represents the demonstration number, listed in the previous section. To create and run a demonstration:

1. Run the `ulcase`***n***`.sql` script corresponding to the demonstration you want to run:

   ```
   $ sqlplus SCOTT/TIGER @ulcasen.sql
   ```

2. Load the demonstration data into the objects:

   ```
   $ sqlldr SCOTT/TIGER ulcasen.ctl
   ```

   The following list provides additional information on the `ulcase2`, `ulcase6`, and `ulcase7` demonstrations:

   - For the `ulcase2` demonstration, you do not have to run the `ulcase2.sql` script.
   - For the `ulcase6` demonstration, run the `ulcase6.sql` script, then enter the following command:

     ```
     $ sqlldr SCOTT/TIGER ulcase6 DIRECT=true
     ```

   - For the `ulcase7` demonstration, run the `ulcase7s.sql` script, then enter the following command:

     ```
     $ sqlldr SCOTT/TIGER ulcase7.ctl
     ```

     After running the demonstration, run the `ulcase7e.sql` script to drop the trigger and package used by this demonstration.

1.8.2 Administering SQL*Loader

SQL*Loader is used by both database administrators and Oracle9*i* users. It loads data from standard operating system files into Oracle database tables.

The SQL*Loader control file includes the following additional file processing option strings, the default being `str`, which takes no argument:

```
[ "str" | "fix n" | "var n" ]
```

Table 1–6 describes the processing options used in the preceding example:

Table 1–6 SQL*Loader Processing Option String

String	Description
`"str"`	Specifies a stream of records, each terminated by a newline character, which are read in one record at a time. This string is the default.
`"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, with the length of each record specified in the first n characters. If you do not specify a value of *n*, SQL*Loader assumes a value of 5.

If you do not select the 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 scan for record terminators.

1.8.2.1 Newline Characters 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 character, include the length of the newline (one character) when specifying the record length to SQL *Loader.

For example, to read the following file, specify `fix 4` instead of `fix 3` to include the additional newline character:

```
AAA<cr>
BBB<cr>
CCC<cr>
```

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 character, terminate all records with a newline character.

Caution: Certain text editors, such as edt, 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.

1.8.2.2 Removing Newline Characters

Use the `position(x:y)` function in the control file to discard the newline characters from fixed length records rather than loading them. For example, enter the following in your control file to discard newline characters from the fourth position:

```
load data
infile xyz.dat "fix 4"
into table abc
( dept position(01:03) char )
```

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

1.8.3 PL/SQL Demonstrations

PL/SQL includes a number of sample programs that you can load. The Oracle9*i* database must be open and mounted to work with the sample programs.

1.8.3.1 PL/SQL Kernel Demonstrations

The following PL/SQL kernel demonstrations are available:

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

To build and run the PL/SQL kernel demonstrations, enter the following commands:

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

```
$ set defa ora_root:[PLSQL.demo]
$ sqlplus scott/tiger
```

2. To load the demonstrations, enter the following command:

```
SQL> @examp<n>.sql
```

> **Note:** Build the demonstrations as any Oracle user with sufficient permissions. Run the demonstrations using the same oracle user account.

To run the `extproc` demonstration:

1. Add the following lines to the `tnsnames.ora` file:

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

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

```
SID_LIST_LISTENER=(SID_LIST=(SID_DESC= -
(SID_NAME:extproc)(program=PIST03:[xyz.network.admin]extproc)))
```

3. From SQL*Plus, enter:

```
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> GRANT CREATE LIBRARY TO SCOTT;
Grant succeeded.
SQL> CONNECT SCOTT/TIGER
Connected.
SQL> CREATE LIBRARY DEMOLIB AS 'ora_root:[bin]extproc.exe';
Library created.
```

4. To run the demonstration, enter:

```
SQL> @extproc
```

1.8.3.2 PL/SQL Precompiler Demonstrations

The following precompiler demonstrations are available:

```
examp9.pc
examp10.pc
sample5.pc
sampel6.pc
```

To build a single demonstration, follow the steps shown below for the `examp9` example.

```
$ proc examp9.pc
$ define rt_includes ora_progint_incl,ora_progint_vms_hdrs,ora_progint_decc_hdrs
$ CC/nowarn/include=rt_includes examp9.c
$ lnproc examp9
```

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

```
$ Run examp9
```

2

Tuning Oracle9*i* on HP OpenVMS Alpha

The more your Oracle9*i* applications increase in complexity, the more you must tune the system to optimize performance and prevent bottlenecks. This chapter describes how to configure yourOracle9*i* installation to optimize its performance.

It contains the following sections:

2.1 Importance of Tuning

Oracle9*i* is a highly-optimizable software product. Frequent tuning optimizes system performance and prevents bottlenecks. Although this chapter is written from the perspective of single-instance, most of the performance tuning tips provided here are also valid when using Oracle9*i* Real Application Clusters.

> **See Also:** For more information on tuning, refer to the *Oracle9i Real Application Clusters Concepts and Administration* and *Oracle9i Database Performance Tuning Guide and Reference*.

2.1.1 Types of Performance Bottlenecks

Performance bottlenecks are often caused by the following:

- Memory contention

  Memory contention occurs when processes require more memory than is available. When this occurs, the system pages and swaps processes between memory and disk.

- Disk I/O contention

  Disk I/O contention is caused by poor memory management, poor distribution of tablespaces and files across disks, or a combination of both.

- CPU contention

  Although the OpenVMS kernel usually allocates CPU resources effectively, many processes compete for CPU cycles and this can cause contention. If you installed Oracle9*i* in a multiprocessor environment, there might be a different level of contention on each CPU.

- Oracle resources contention

  Contention is also common for Oracle resources such as locks and latches.

2.2 Oracle SQL Tuning Tools

Oracle provides a variety of tools for tuning SQL, including the V$ performance views, the EXPLAIN PLAN command, the SQL TRACE facility, the TKPROF facility, the Autotrace report and the STATSPACK.

> **See Also:** the *Oracle9i Database Performance Tuning Guide and Reference* for information on how to use the SQL tuning tools.

2.3 Tuning Memory Management

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

The Oracle buffer manager ensures that the more frequently accessed data is cached longer. If you monitor the buffer manager and tune the buffer cache, you can have a significant influence on Oracle9*i* performance. The optimal Oracle9*i* buffer size for your system depends on the overall system load and the relative priority of Oracle over other applications.

2.3.1 Allocate Sufficient Swap Space

Try to minimize swapping because it causes significant OpenVMS overhead. To check for swapping, enter the following command:

```
$ show memory/files
```

If your system is swapping and you must conserve memory:

- Avoid running unnecessary system daemon processes or application processes.
- Decrease the number of database buffers to free some memory.

> **See Also:** the operating system documentation for more information on the `$ show memory` command.
>
> To install additional Page and Swap files, refer to the VMS INSTALL Utility. For additional comprehensive information on managing Page and Swap files refer to the *OpenVMS System Manager's Guide*.

2.3.2 Control Paging

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

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

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

- Install more memory.
- Move some of the work to another system.
- Decrease the number of database buffers to free some memory.

2.3.3 Adjust Oracle Block Size

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

The DB_BLOCK_SIZE initialization parameter sets the database block size database-wide.

To see the current value of the DB_BLOCK_SIZE parameter, or to create new tablespaces with a different size, enter the SHOW PARAMETER command in SQL*Plus.

2.4 Tuning Disk I/O

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

2.5 Monitoring Disk Performance

To monitor disk performance, use the `$Monitor Disk` command.

The average value from `$monitor disk/item=queue` should not exceed 0.8-1.0. If it does or if the MAX value is high, you suffer from an I/O bottleneck.

2.6 Tuning CPU Usage

This section provides information on tuning CPU usage.

2.6.1 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 priority levels causes unexpected effects on contention and response times. Oracle does not support changing the priority of user and background processes.

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 may suffer poor response time.

2.7 Using Trace and Alert Files

This section describes the trace (or dump) and alert files that Oracle9*i* creates to diagnose and resolve operating problems.

2.7.1 Trace Files

Each server and background process can write to an associated trace file. When a process detects an internal error, it writes information on the error to its trace file. The filename format of a trace file is:

```
nodename_sid_processtype_processname_number.trc,
```

where:

nodename is the name of the machine where an instance is running, and

sid is the instance system identifier,

processtype is the FG (foreground) or BG (background),

processname is the process that generates it; and

number is a unique 3 digit numeric identifier.

A sample trace filename is:

```
ORA_ROOT:[ADMIN.<dbname>.BDUMP]NODEA_PROD_BG_PMON_002.TRC
```

All trace files for background processes are written to the destination directory specified by the BACKGROUND_DUMP_ DEST initialization parameter. All trace files for server processes are written to the destination directory specified by the USER_DUMP_DEST initialization parameter.

Set the MAX_DUMP_FILE_SIZE initialization parameter to at least 10000 to ensure that the trace file is large enough to store error information.

2.7.2 Alert Files

The `<nodename>_sid_alert.log` file stores significant database events and messages. Anything that affects the database instance or global database is recorded in this file. This file is associated with a database and is located in the directory specified by the BACKGROUND_DUMP_DEST initialization parameter.

3

Setting Up Oracle9*i* Users

This chapter assumes that you have installed Oracle9*i* and have created a database and started an instance. This chapter describes how to set up Oracle9*i* users, as well as how to set up certain application development tools. Only those tools having special requirements are described.

This chapter contains the following topics:

3.1 Granting Access to Oracle9*i* Users

When Oracle9*i* is installed with the Group option, all Oracle9*i* users must belong to the same group that includes the Oracle9*i* account. If Oracle9*i* is installed with the SYSTEM option, it can be accessed by any user in any UIC group

To grant users access to Oracle9*i*, complete the following steps:

1. Include the following line in each user's LOGIN.COM file to identify that user's default instance and database:

   ```
   $ @device:[directory]orauser sid
   ```

   where:

   device is the disk name or logical name where the Oracle software resides; and

   directory is the directory path to the location where the Oracle software resides; and

   sid is the SID of the Oracle9*i* instance that the user should access. This is a qualifier that is optional, depending on the circumstances of the instance setup; and

   For example:

   ```
   $ @DISK$DISK1:[ORACLE9i]ORAUSER PROD
   ```

   Instead of completing step 1, you can define a symbol in the system-wide login procedure (typically, SYLOGIN.COM) that executes a particular ORAUSER.COM file. This method might be more useful if users access multiple instances, and therefore need to execute a database-specific ORAUSER file with the proper parameters.

   For example:

   ```
   $ PROD=="@DISK$DISK1:[ORACLE9I]ORAUSER PROD"
   ```

2. Ensure that each user's OpenVMS account meets at least the minimum requirements for ASTLM, BYTLM, ENQLM, WSDEFAULT, WSEXTENT, WSQUOTA, and PGFLQUO.

   For more information about account quotas, refer to the "Setting Up the Oracle Accounts" chapter of the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha*.

3. Create the Oracle9*i* user accounts with the CREATE USER and ALTER USER commands. Use the GRANT command to grant the appropriate database privileges or roles as documented in the *Oracle9i Database Administrator's Guide.*
4. If you have a user who uses the SQL*PLUS utility to start up or shut down an Oracle9*i* instance, use the OpenVMS utility AUTHORIZE to add an ORA_<sid>_DBA or ORA_DBA process rights identifier to the user's OpenVMS account from the OpenVMS rights database. For more information, refer to the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha*.

3.2 Ending a User's Session

You can end a user's Oracle9*i* session as follows:

- Issue the ALTER SYSTEM KILL SESSION command in SQL*Plus.

Note that the session is flagged for KILL but remains open until the user issues another command. Only then will the user's session actually terminate.

4

Starting Up and Shutting Down Oracle9*i*

This chapter describes different ways to start up or shut down Oracle9*i*. These methods include using STARTUP and SHUTDOWN files, or using SQL*Plus.

This chapter contains the following topics:

4.1 Starting Up Oracle9*i*

Before you can start Oracle9*i*, both an instance and a database must exist on your local system. If you did not install Oracle9*i*, consult the person who did.

This section presents the following topics:

4.1.1 Before Start Up

If you rebooted your HP OpenVMS Alpha system (for example, due to a system crash), you should read this section. If not, you can skip this section.

After rebooting OpenVMS, you must perform the following step before starting Oracle9*i*:

Run the ORAUSER.COM file, specifying the full directory path. For example:

```
$ @DISK$A31:[MYROOT]ORAUSER.COM
```

Note that when the DBA executes this file, a check will be performed to determine whether `oracle_home:insoracle.com` needs to be executed; if so, that file is called. This file installs the shared global sections that make a shareable ORACLE image known to the system.

The following images are installed:

ORACLIENT.EXE

ORACLE.EXE

4.1.2 Starting Oracle9*i* via SQL*Plus

You can also start an instance of Oracle9*i* using SQL*Plus. Refer to the instructions in this manual on setting up SQL*Plus on your HP OpenVMS Alpha platform. Refer to the generic (platform-independent) Oracle Server documentation for instructions on using SQL*Plus.

You might choose to complete startup tasks separately when monitoring instance performance, for example, or you might want to start an instance and open a database after making some modifications.

4.1.2.1 Identifying the Current Instance

When starting up Oracle9*i*, you start up the current instance. The current Oracle9*i* instance is identified by the value of the logical name ORACLE_SID. For example, if the value of ORACLE_SID is currently V9, the current instance is the instance with the SID V9. If you have not reassigned the ORACLE_SID logical name, the value of ORACLE_SID is the SID specified during installation. To change the current instance before starting the Oracle9*i* with SQL*Plus, you should run the ORAUSER.COM with the appropriate `SID` as parameter.

If ORACLE_SID is undefined or incorrect, you receive the following error:

```
ORA-07582,  spstp: ORA_SID has an illegal value.
```

4.1.2.2 Specifying Startup Parameters

When the current Oracle9*i* instance is started, the SGA is created and initialized with the startup parameters set in the distributed parameter file, INIT.ORA, in the ORA_DB directory. When using SQL*Plus, you can use another startup file that sets different parameter values by including the PFILE option with the STARTUP command to identify an alternative parameter file. If the file is not in the current default directory, you must include the directory location of the file:

```
SQL> STARTUP PFILE=INIT<sid>.ORA
```

4.1.2.3 Starting the Server using SQL*Plus

To start Oracle9*i*, you must have the process rights identifier ORA_DBA or ORA_<sid>_DBA assigned to your user account in the OpenVMS rights database and you must run the .COM file that makes the logical name assignments required to run Oracle9*i*.

Before starting up Oracle9*i*, run the ORAUSER.COM file to set the desired instance.

After running the above .COM file, run SQL*Plus and execute the appropriate STARTUP command(s), as documented in the *Oracle9i Database Administrator's Guide*. You can issue the single SQL*Plus command, STARTUP, or execute the three separate SQL*Plus commands documented in the *Oracle9i Database Administrator's Guide* to start the Oracle9*i* Enterprise Edition.

The SQL*Plus command STARTUP starts the current ORACLE instance, creating the SGA in OpenVMS shared memory and creating the detached processes. It then mounts the database and opens it.

4.1.3 Starting Oracle9*i* Remotely via SQL*Plus from an OpenVMS Client

You can use SQL*Plus on an OpenVMS client to start up an Oracle9*i* database instance on a remote HP OpenVMS Alpha system.

4.1.4 Steps to Perform on Remote System Database

The following steps must be performed on the remote system where the database resides:

1. Create a password file using ORAPWD. The password file can be either exclusive or shared. For this example, we will assume an exclusive password file. The syntax for ORAPWD is as follows:

   ```
   $ ORAPWD FILE=<fname> PASSWORD=<password> ENTRIES=<users>
   ```

2. Define a system logical name to point to the location of the password file. For example:

   If using an exclusive password file:

   ```
   $ DEFINE/SYSTEM/EXEC ORA_<sid>_PWFILE -
     ddcn:[directory]<fname>
   ```

   If using a shared password file:

   ```
   $ DEFINE/SYSTEM/EXEC ORA_PWFILE -
     ddcn:[directory]<fname>
   ```

3. Edit `INIT<sid>.ORA` and add the following line:

   If using an exclusive password file:

   ```
   REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE
   ```

   If using a shared password file:

   ```
   REMOTE_LOGIN_PASSWORDFILE = SHARED
   ```

4. Stop and restart the database instance.
5. Copy `INIT<sid>.ORA` from the server to any directory on the client.

4.1.5 Steps to Perform on Client System

The following steps must be performed on the client system from which the database is to be started:

1. Ensure that there is a `TNSNAMES.ORA` entry for the `SID` on the remote system where the database resides.
2. Define the process logical name `ORA_DFLT_HOSTSTR` to the `Oracle Net V9 ALIAS` for the remote system. For example:

```
$ DEFINE ORA_DFLT_HOSTSTR <Oracle Net V9 alias>
```

3. Define the process logical name that points to the complete file specification for the `INIT` file copied in Step 5 above. For example:

```
$ DEFINE ORA_PARAMS -
  ddcn:[directory]INIT.ORA
```

4. Invoke `SQL*PLUS` and issue the commands as follows. When prompted for the password, enter the password specified in Step 1 above (server side) when the password file was created.

```
$ sqlplus "/ as sysdba"
SQL*Plus: Release 9.2.0.1.0 - Production on Mon Aug 12 04:36:26 2002

(c) Copyright 2002 Oracle Corporation.  All rights reserved.
Connected to an idle instance.
SQL> startup
ORACLE instance started.

Total System Global Area  555189984 bytes
Fixed Size                   730848 bytes
Variable Size             520093696 bytes
Database Buffers           33554432 bytes
Redo Buffers                 811008 bytes
Database mounted.
Database opened.
SQL>
```

5. At this point, the remote database is up and running.

4.1.6 Starting Oracle9*i* Remotely via SQL*Plus from a Windows PC Client

The following steps must be performed on the remote system where the database resides.

4.1.7 Steps to Perform on Remote System Database

1. Create a password file using ORAPWD. The password file can be either exclusive or shared. For this example, we will assume an exclusive password file. The syntax for ORAPWD is as follows:

   ```
   $ ORAPWD FILE=<fname> PASSWORD=<password> ENTRIES=<users>
   ```

2. Define a system logical name to point to the location of the password file. For example:

   If using an exclusive password file:

   ```
   $ DEFINE/SYSTEM/EXEC ORA_<sid>_PWFILE -
     ddcn:[directory]<fname>
   ```

   If using a shared password file:

   ```
   $ DEFINE/SYSTEM/EXEC ORA_PWFILE -
     ddcn:[directory]<fname>
   ```

3. Edit INIT<sid>.ORA and add the following line:

   If using an exclusive password file:

   ```
   REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE
   ```

   If using a shared password file:

   ```
   REMOTE_LOGIN_PASSWORDFILE = SHARED
   ```

4. Stop and restart the database instance.
5. Copy INIT<sid>.ORA from the server to any directory on the client.

The following steps must be performed on the client system from which the database is to be started:

6. Ensure that there is a TNSNAMES.ORA entry for the SID on the remote system where the database resides.

7. Invoke SQL*Plus from File Manager (Windows 3.x) or Windows Explorer (Windows95/98/2000/NT). It should be located in the following directory: `\ORAWIN\BIN` for Windows 3.x, `\ORAWIN95\BIN` for Windows95, and `\ORANT\BIN` for Windows NT. When prompted for the password, enter the password specified in Step 1 above (server side) when the password file was created. `net9_V9_ALIAS` is the `TNSNAMES.ORA` alias for the remote database.

```
Oracle SQLPLUS Release 9.2.0.1.0 - Production
(c) Copyright 2002, Oracle Corporation.  All Rights Reserved.
Oracle9i Enterprise Edition Release 2 (9.2.0.1.0) - Production
PL/SQL Release 2 (9.2.0) - Production
SQL> connect @<net9_V9_alias>
Password:
SQL> startup pfile=<DOS path to INIT.ORA>
ORACLE instance started.
Total System Global Area              555189984 bytes
Fixed Size                               730848 bytes
Variable Size                         520093696 bytes
Database Buffers                       33554432 bytes
Redo Buffers                             811008 bytes
Database mounted.
Database opened.
SQL> exit
```

8. At this point, the remote database is up and running.

4.2 Shutting Down Oracle9*i*

This section describes the following three methods for shutting down Oracle9*i*:

- Shutting Down Oracle9i using SQL*Plus
- Stopping Oracle Users Before Database Shutdown
- De-installing Shareable Images

After all instances on a node have been shut down, you may de-install the shareable images. Refer to "De-installing Shareable Images" on page 4-9 for information on de-installing the shareable images.

4.2.1 Shutting Down Oracle9*i* using SQL*Plus

You can shut down an instance of Oracle9*i* using SQL*Plus. Refer to Chapter 7, "Administering SQL*Plus" in this manual for instructions on setting up SQL*Plus on your OpenVMS platform. Then, refer to the generic (platform-independent) Oracle Server documentation for instructions on using SQL*Plus.

4.2.2 Stopping Oracle Users Before Database Shutdown

The command SHUTDOWN IMMEDIATE may hang if you have persisting connections in the environment; for example, if you are running processes associated with monitoring tools such as OEM Agent. In that case, it is advisable to terminate the connections from the operating system level by issuing the following command:

```
"$ stop/id=<pid>"
```

For example:

```
SQL> select sid,serial#,process from v$session;
```

SID	SERIAL#	PROCESS
1	1	20C0018B
2	1	20C0018C
3	1	20C0018D
4	1	20C0018E
5	1	20C0018F
6	1	20C002DD

```
SQL>host stop/id=20C0018B
SQL>host stop/id=20C0018C,
```

and et cetera.

You are strongly discouraged from stopping the sessions within ORACLE by using the ALTER SYSTEM KILL SESSION command followed by a `stop/id` on the operating system level.

When you issue an ALTER SYSTEM KILL SESSION command, it marks the process for deletion by PMON. If you then kill the process before PMON can get to it, confusions results and a clean process deletion does not occur. The deleted process appears to still be connected. Thus, the SHUTDOWN IMMEDIATE hangs and the

partially dead process can't respond to the logoff command issued by the SHUTDOWN.

Use either ALTER SESSION KILL SESSION or HOST STOP/ID=. Don't use both.

4.2.3 De-installing Shareable Images

After shutting down all Oracle9*i* instances on a node, if you intend to do a machine shutdown or code relink, remove the shareable images by issuing the following command:

```
$ REMORACLE
```

5

Managing the Database

Ensuring that Oracle9*i* operates successfully can involve tuning the system or modifying parameters. These tasks require a thorough understanding of HP OpenVMS Alpha system administration as well as the concepts documented in the *Oracle9i Database Administrator's Guide*.

This chapter contains the following topics:

5.1 SQL*Plus and Oracle Net

When you start up SQL*Plus, a bequeath protocol adapter connection will be made if no TNS connect descriptor is supplied. Refer to Chapter 9, "Oracle Net on HP OpenVMS Alpha" for more information about bequeath adapter.

5.2 Creating Multiple Control Files

Three control files are created whenever you create a database. by default, the files are named `control01.ctl`, `control02.ctl` and `control03.ctl`. They reside in the directory pointed to by the logical `ora_db`. However, Oracle Corporation recommends that you back up the control files and create additional copies. When you add more control files, be sure to add the new filenames and locations to the CONTROL_FILES initialization parameter.

Refer to the *Oracle9i Database Administrator's Guide* for general information. Specific information for HP OpenVMS Alpha can be summarized as follows:

- By default, the control files reside in the ORA_DB directory.
- Control files can be moved to any location.
- To guard against device failure, the control files should be placed on separate devices.

5.3 Managing Database Files

During the ORACLE installation procedure, you create one database file in the directory referenced by the logical name ORA_DB, typically `ORA_ROOT:[oradata.<dbname>]system01.dbf`.

To add database files to an existing tablespace, use the SQL statement ALTER TABLESPACE. You cannot remove or delete a file; however, you can remove tablespaces other than the SYSTEM tablespace.

5.3.1 Using Commands to Manage Database Files

There are some commands that are useful in managing database files. The commands mentioned here are documented fully in the *Oracle9i Database Administrator's Guide*.

5.3.1.1 ALTER DATABASE

In addition to using the ALTER DATABASE command to mount, open, or close a database, to add or drop redo log files, and to archive redo log files, this command can be used to rename and/or move tablespace files and redo log files.

You cannot use the ALTER DATABASE BACKUP CONTROLFILE command to back up control files to tape. To back up control files to tape, back up to disk and then copy to tape.

5.3.1.2 DROP TABLESPACE

Before using the DROP TABLESPACE INCLUDING CONTENTS command, take the tablespace offline to ensure that no temporary segments are in use.

5.3.2 Adding Files

When specifying files to be added to the database, logical names are fully translated to either physical device names or system-level concealed logical names (if defined) and then written to the control file.

5.3.3 Renaming Files

If the name of the physical device is somehow disassociated with the database file location(s), the RDBMS cannot access these files. Use the ALTER DATABASE command to RENAME the file to its current location. After renaming the files, shut down the database and then back up the control files as in the following example:

```
SQL> ALTER DATABASE RENAME FILE
2> 'DISK$1:[ORACLE9i.oradata.V9TEST]SYSTEM01.DBF' TO
3> 'MY$DISK:[ORACLE9i.oradata.V9TEST]SYSTEM01.DBF'
SQL> EXIT
$ BACKUP/LOG/VERIFY/IGNORE=INTERLOCK -
DISK$1:[ORACLE9i.oradata.V9TEST]*.CTL -
MY$DISK:[ORACLE9i.oradata.V9TEST]*.CTL
```

Note: The physical device name and the file location must appear exactly as in the control file. Enter the following commands to get the physical device name and the database file location(s):

```
$ SQLPLUS/NOLOG
SQL> CONNECT / AS SYSDBA
SQL> SELECT * FROM V$DBFILE;
SQL> DISCONNECT
```

5.3.4 Moving Tablespace Files

To move a tablespace file to a new location perform the following steps:

1. Identify and write down the exact, fully qualified filename from the data dictionary view and shut down the database. The physical device name and the file location must appear exactly as in the control file and the data dictionary view, DBA_DATA_FILES or V$LOGFILE.

```
$ SQLPLUS/NOLOG
SQL> CONNECT / AS SYSDBA
SQL> SELECT * from V$DBFILE;
SQL> SELECT * from V$LOGFILE;
SQL> SHUTDOWN
SQL> EXIT
```

2. Back up the tablespace files that you want to move as well as the control files.
3. Copy or move the file to a new location (use BACKUP/VERIFY/DELETE to move the file).

```
$ BACKUP/IGNORE=NOBACK/DELETE/VERIFY -
<device>:[<dir>]<filename>.<ext> -
<new_device>:[<new_dir>]<new_filename>.<ext>
```

4. Without opening it, mount the database in exclusive mode.

```
$ SQLPLUS/NOLOG
SQL> CONNECT / AS SYSDBA
SQL> STARTUP EXCLUSIVE MOUNT <dbname>
```

5. Rename the file in the database using the exact string taken from V$dbfile.

```
SQL> ALTER DATABASE
   2> RENAME FILE '<device>:[<dir>]<filename>.<ext>'
   3> to '<new_device>:[<new_dir>]<new_filename>.<ext>';
SQL> ALTER DATABASE <dbname> OPEN;
SQL> EXIT
```

6. Back up the control files.

5.3.5 Moving Redo Log Files

Perform the following steps to move a redo log file to a new location:

1. Identify the exact, fully qualified filename of the redo log files that you want to move by one of the following methods:

   a. If your database instance is up, issue the following query:

```
SQL> SELECT * FROM V$LOGFILE;
```

2. Shut down the database, make a backup copy of the redo log files in the new location, and mount the database in exclusive mode (not opened).

Note: After the database is shut down, make image copies of all database, control, and redo log files as a precaution against any problems that can arise during this procedure.

```
$ SQLPLUS/NOLOG
SQL> CONNECT / AS SYSDBA
SQL> SHUTDOWN
SQL> EXIT
$ BACKUP/IGNORE=NOBACK -
<old_device>:[<dir>]<filename>.<ext> -
<new_device>:[<new_dir>]<new_filename>.<ext>
$ SQLPLUS/NOLOG
SQL> CONNECT / AS SYSDBA
SQL> STARTUP EXCLUSIVE MOUNT <dbname>
```

Note: Having the database mounted and closed is essential when working with the redo log files. This prevents any log files from becoming online or marked as current by the LGWR.

3. From SQL*Plus, rename the files in the database using the ALTER DATABASE command. Specify the full file path.

```
SQL> CONNECT / AS SYSDBA
SQL> ALTER DATABASE RENAME FILE
   2> '<device>:[<dir>]<old_redofile1>.RDO',
   3> '<device>:[<dir>]<old_redofile2>.RDO' to
   4> '<device>:[<dir>]<new_redofile1>.RDO',
   5> '<device>:[<dir>]<new_redofile2>.RDO';
```

   The filenames specified must be correct and the new files must already exist. If either of these requirements are not met, the statement will fail.

4. Shut down the database using the following command:

```
SQL> SHUTDOWN
```

5. Back up the control files for safety.

6. Restart the database using the following commands.

```
SQL> CONNECT / AS SYSDBA
SQL> STARTUP OPEN <dbname>
SQL> EXIT
```

5.4 Database Verification Utility and Other Useful Utilities

This section gives information about the following:

- Database verification utility
- Other useful utilities

5.4.1 Database Verification Utility

The database verification utility (DBV) is the preferred technique for verifying the integrity of your database. Invoke this utility with the `DBV` symbol on OpenVMS.

To use this utility to verify data in an Oracle9*i* Release 2 (9.2.0.2) database, point to the 9.2 files from your Oracle9*i* Release 2 (9.2.0.2) installation.

Additional Information: Refer to the *Oracle9i Database Utilities* manual. As this document mentions, SQL*Plus can also be used to verify your database.

5.5 Debugging Database Processes with ORAMBX

Sometimes an Oracle server process will seem to be spinning or hung. To provide Oracle with useful information on this process, you may occasionally be asked to generate a trace file containing debugging information.

The command utility ORAMBX is one way to obtain this information. ORAMBX takes the process name, not its PID, as an argument. At the prompt, you feed one command at a time. When you have finished sending commands, exit with control-Z. Then a trace file will exist in ORA_DUMP that contains information that is useful for debugging purposes.

Table 5–1 lists the most common ORAMBX commands and their functions:

Table 5–1 ORAMBX Commands

Command	Function
DUMP	Dump call stack
PGA	Dump the fixed pga
SGA	Dump the fixed sga
SYSTEM	Perform system state dump
EVENT	Set process event
SESEVENT	Set session event
BLOCK	Dump block(s) at specified level
MEMORY LOG	Dump log of memory protection events
SUSPEND	Suspend process at current mode
FLUSH	Flush any pending writes to trace file

The two most useful commands are DUMP and SYSTEM. DUMP shows the process' call stack, which is useful if the process is hanging or spinning. The command DUMP 1 simply generates a printout of the call stack. The command DUMP 10 prints a call stack and information about all cursors, queries, and other Oracle process information available. Likewise, the command SYSTEM 1 produces a small amount of interesting information about an instance, while SYSTEM 10 tells about almost anything happening in the instance, processes, cursors, locks.

Note: In a client-server situation, the ORAMBX commands can only be issued to the server-process, which is where the work is performed. Running ORAMBX against the client application will result in an error from ORAMBX. This is normal.

6

Backing Up and Archiving Your Database

If the server is interrupted by a hardware failure, an operating system error, or an unexpected process termination, the result can be damaged files or a database that contains inconsistent data. Recovery is then needed to reconstruct the database in such a way that no committed transactions are lost and no uncommitted changes are retained.

This chapter describes the procedures for backing up the database. You must complete database backups periodically to be able to recover data if you have a media failure.

This chapter contains the following topics:

6.1 Archiving Redo Log Files

How much of the database you can recover if media failure occurs depends upon whether you archive the redo logs and how often you back up and export the database. Refer to the *Oracle9i Database Administrator's Guide* for more information on archiving.

Information in the redo logs is always sufficient to guarantee instance recovery, regardless of the mode in which the logs are used. However, full media recovery is possible only if you use ARCHIVELOG mode and archive in offline files. If you use NOARCHIVELOG mode, be sure to shut down Oracle9*i* before backing up the database.

When a redo log file has filled, a checkpoint occurs. Additional checkpoints can be triggered by reducing the value of the INIT.ORA parameter LOG_CHECKPOINT_INTERVAL. Each checkpoint guarantees that information in the redo log file is written to the database. Frequent writes can speed recovery, because there will be less data in the logs to reapply to the database.

Three initial redo logs of 100 MB each are created during the installation procedure; you can create additional logs with the ALTER DATABASE command. These initial logs are created in NOARCHIVELOG mode; you can change them to ARCHIVELOG mode with the ALTER DATABASE command. To see the current status of your log files, use the command ARCHIVE LOG LIST. Refer to the *Oracle9i Database Administrator's Guide* for more information.

Note: When running in RAC mode, the redo logs for all instances must be archived, or none at all. The ARCHIVELOG keyword of the ALTER DATABASE command affects the entire database, not just the current instance, and must only be issued while the database is mounted in exclusive mode.

6.1.1 Specifying Archive Destinations

You can archive redo log files to disk. If you wish to archive redo logs to tape, you must first archive them to disk, and then use the OpenVMS BACKUP utility to copy them from disk to tape. You should never archive directly to tape. Refer to Hewlett-Packard's document *OpenVMS Guide to Tapes and Devices*, and to the *Oracle9i Database Administrator's Guide* for more information.

To specify a disk file as the archive destination, use the following conventions:

```
LOG_ARCHIVE_DEST   =  <diskname>:[<directory_name>]
LOG_ARCHIVE_FORMAT =  <filename>
```

You must specify a full file name or valid file name format using the variables. This file name is appended to the LOG_ARCHIVE_DEST string to create the archived redo log files in the specified location.

Note: The value for LOG_ARCHIVE_FORMAT is *not* enclosed in single quotes on OpenVMS.

All references to LOG_ARCHIVE_DEST must be accompanied by LOG_ARCHIVE_FORMAT and the statements modified appropriately. For example:

```
LOG_ARCHIVE_DEST   = DISK$ARC:[ORACLE.V9.ORADATA.PROD]
LOG_ARCHIVE_FORMAT = MIS_SEQ%s_SCN%c.ARC
```

For faster crash recovery, the following archive log naming convention is recommended:

```
LOG_ARCHIVE_FORMAT =  Name_THR%t_SEQ%s_SCN%c.ARC
```

The disk name, directory name, and prefix for the archived redo log files are specified in this destination command string. The prefix is added to all the redo log files names that are archived.

6.1.2 Archiving Automatically

If a database is running with ARCHIVELOG mode enabled, the redo log files of a given instance must be archived manually or automatically. If the database is also mounted in RAC mode, some instances can be archived manually, while others are archived automatically, as long as all instances have their redo log files archived.

To archive redo logs automatically, dedicate a disk drive without any other ORACLE files for archiving your files and then complete the following steps:

1. Shut down the current instance.
2. Set the value of the LOG_ARCHIVE_START parameter in the INIT.ORA file to TRUE.

3. Specify the destination of the archived files with the LOG_ARCHIVE_DEST parameter in the same parameter file (either the instance-specific INIT<sid>.ORA file, or INIT.ORA itself).
4. Restart the instance.
5. If the database is mounted in RAC mode, and you want other instances to archive automatically, repeat the steps above, skipping step, if you added the LOG_ARCHIVE_START and LOG_ARCHIVE_DEST parameters to INIT.ORA (rather than the current instance's INIT<sid>.ORA parameter file).

You can also enable automatic archiving for a database instance that is running in ARCHIVELOG mode without changing INIT<sid>.ORA by using the SQL*Plus command ARCHIVE LOG as in the following command:

```
SQL> ARCHIVE LOG START <filename>
```

The next time an online redo log file needs to be archived for the current instance, it will be archived automatically until the instance is next shut down. To make archiving permanent, you must set the LOG_ARCHIVE_START, LOG_ARCHIVE_DEST, and LOG_ARCHIVE_FORMAT parameters in the appropriate parameter file (INIT.ORA or the instance's *<setup_node>*_*<sid>*_INIT.ORA parameter file).

When using automatic archiving, errors that occur during archiving and start and stop times of the ARCH process are written either to a trace file in the `ORA_ROOT:[ADMIN.<db_name>.BDUMP]` directory or to the alert log.

6.1.3 Archiving Manually

To archive redo log files for the current instance manually, use the command ARCHIVE LOG. You must specify the log sequence number of the redo log file group to be archived. If you do not specify the archive destination, the destination is derived from the INIT.ORA parameter LOG_ARCHIVE_DEST.

- To archive the first redo log, enter the following command:

  ```
  SQL> ARCHIVE LOG <log_sequence_number> <destination>
  ```

  Replace *<log_sequence_number>* with the number of the log file you want to be archived.

- To archive the next file to be archived, use the NEXT option as in the following command:

```
SQL> ARCHIVE LOG NEXT <destination>
```

- To archive all redo log files, use the ALL option as in the following command:

```
SQL> ARCHIVE LOG ALL <destination>
```

When archiving manually, errors are written to your terminal.

You can also manually archive using the ARCHIVE LOG clause of the ALTER SYSTEM command. The ARCHIVE LOG clause contains all the capabilities of the ARCHIVE LOG command. You can use it to archive the log files of any instance, not just the current instance.

6.2 Backing Up the Database

A database backup is a block-by-block copy of the database files. If you are the DBA, you should backup the database regularly, following the instructions in the following sections:

- Backing Up a Closed Database (offline or cold backup)
- Backing Up an Open Database (online or hot backup)

Both types of backup will restore either all or part of the database to the same condition that existed at the time of backup. To recover any transactions committed after the backup, the DBA must use the redo logs where those transactions were recorded. If you back up files while the database is running, use the redo log files in ARCHIVELOG mode to maintain a record of transactions occurring during the backup.

To back up database files, use the OpenVMS utility BACKUP. The *Oracle9i Database Administrator's Guide* describes the steps for backing up both open and closed databases; when you are ready to complete the step that instructs you to perform the actual backup, run the OpenVMS BACKUP utility.

6.2.1 Backing Up a Closed Database

To back up a closed database, complete the following:

1. Shutdown all instances using the SHUTDOWN NORMAL command.
2. Run the OpenVMS BACKUP utility to copy all database files, redo log files, and control files by entering the following command:

   ```
   $ BACKUP <directory>:<database_filename> -
   [<new_directory>]<new_filename>
   ```

   For example, if your database file is named SYSTEM01.DBF and you are copying to a directory named ARCDIR you would enter the following:

   ```
   $ BACKUP ORA_DB:SYSTEM01.DBF DISK$2:[ARCDIR]SYSTEM01.DBF
   ```

   If you have multiple databases, or if your database files do not reside in the ORA_DB directory, you might need to specify a directory location other than ORA_DB.
3. Restart the instances.

Attention: You can automate much of the backup procedure through the use of scripts.

See the file ORA_RDBMS:READMEVMS.DOC for information about accessing sample scripts.

6.2.2 Backing Up an Open Database

Backing up an open database allows users to have normal access to all online tablespaces during backup.

Note: Do not take the tablespace offline or shut down your system until END BACKUP is completed; the backup might not be useable. If the following warning message occurs during the backup procedure, ignore it and continue with the backup.

```
%BACKUP-W-ACCONFLICT, is open for write by another user
```

To back up an open database, complete the following tasks:

1. Run SQL*Plus, and enter the following command:

   ```
   SQL> ALTER TABLESPACE <tablespace_name> BEGIN BACKUP
   ```

   Specify the name of the tablespace that you want to back up. If you have not created additional tablespaces after installing the database, you can only back up the initial tablespace SYSTEM.

   > **Note:** You must perform this step before proceeding, or else the backup file created in step 2 will be invalid for recovery.

2. Run the BACKUP utility to copy all the database files that make up the tablespace by entering the following:

   ```
   $ BACKUP/IGNORE=(INTERLOCK,NOBACKUP)-
   ORA_DB:<database_filename> -
   [<new_directory>]<new_filename>
   ```

   If you have multiple databases, or if your database files do not reside in the ORA_DB directory, you might need to specify a directory location other than ORA_DB.

3. Run SQL*Plus and enter the following command:

   ```
   SQL> ALTER TABLESPACE <tablespace_name> END BACKUP;
   ```

   > **Note:** The BEGIN BACKUP and END BACKUP are vital. Your backups will be unusable if these commands are not used in the steps listed above.

   Repeat steps 1 - 3 for all tablespaces you want to back up.

6.2.3 Backing Up Data Structures and Definitions

A database backup is a physical copy of a database. To copy the data structures and data definitions in a database in a logically organized format, you must use the Export utility. Normally, you will need a logical copy of the database when a user has dropped a table and you want to restore only that table. Exports also allow selective recovery and let you transfer a single user's data or a specific set of tables. If a user accidentally drops a table, you can recover the table from an export. Image backups do not provide this flexibility.

Note: Import/Export messages go to SYS$ERROR, not SYS$OUTPUT and can be saved to file if you use the LOGFILE option.

You can export the entire database or portions of the database. You can also perform incremental exports, which save only tables that changed since the last export; these exports are quicker and more convenient. To recover the export file generated by the Export utility, use the Import utility. For information about using these utilities, refer to *Oracle9i Database Utilities*.

Note that under OpenVMS, you can copy export files to tape if you specify a block size of 4096 bytes.

6.2.3.1 Exporting to Other OpenVMS Machines

To export files to tape for transfer to another OpenVMS machine, use the following procedure:

```
$ ALLOCATE <tape_device_name>
$ INIT <tape_device_name> <tape_label>
$ MOUNT/BLOCKSIZE=<recordlength> <tape_device_name> -  <tape_label>
$ EXP <username/password>
```

Several prompts appear at this point; respond as appropriate. When prompted to supply the name of the Export file, use the following form:

```
EXPORT FILE:EXPDAT.DMP > : <tape_device_name>:EXPDAT.DMP
```

When the Export session has completed, enter the following commands:

```
$ DISMOUNT <tape_device_name>
$ DEALLOCATE <tape_device_name>
```

6.2.3.2 Exporting to Non-OpenVMS Machines

To export files to tape for transfer to a non-OpenVMS machine, enter the following commands:

```
$ ALLOCATE <tape_device_name>
$ INIT <tape_device_name> <tape_label>
$ MOUNT/FOREIGN/BLOCKSIZE=<recordlength> <tape_device_name>
$ EXP <username/password>
```

Several prompts appear at this point; respond as appropriate. When prompted to supply the name of the Export file, use the following form:

```
EXPORT FILE:EXPDAT.DMP > : <tape_device_name>:EXPDAT.DMP
```

When the Export session has completed, enter the following commands:

```
$ DISMOUNT <tape_device_name>
$ DEALLOCATE <tape_device_name>
```

Suggestion: If you want to create an export file and move it between systems via FTP, you should use binary mode and set RECORDLENGTH to 512.

6.3 Exporting to and Importing from Multiple Tapes

This section describes how to export to and import from multiple tapes. It is a good idea to have a copy of files stored on tapes.

You must have the OPER privilege to perform the following tasks. Additionally, issue the command REPLY/ENABLE=TAPES. This command directs the output to your terminal rather than the operator's console.

6.3.0.3 Exporting with Multi-Reel Files

Multi-reel export files are only possible for OpenVMS tapes; that is, tapes not mounted with the FOREIGN option. The ANSI standard format used by OpenVMS for tapes mounted FOREIGN does not define multi-reel volumes. You can usually work around this limitation of ANSI format using user-level or table-level exports.

6.3.1 Exporting to Multiple Tapes

To export to multiple tapes, enter the following commands:

```
$ INIT <tape_device_name> <tape_label>
$ MOUNT/BLOCK=4096 <tape_device_name> <tape_label>
$ EXP <username>/<password>
```

At this point the export starts and you are prompted to enter the export filename as in the following example:

```
Export file:EXPDAT.DMP > <tape_device_name>:<filename>
```

The export proceeds to the end of the reel.

In the computer room where the tapes are kept perform the following steps:

1. Make sure a tape drive is allocated.
2. The tape rewinds and dismounts by itself.
3. A message flashes onto the operator's terminal instructing to mount the second tape. A request number is provided.
4. The operator mounts the next tape and enters the following statement:

   ```
   $ REPLY/TO=<request_number>
   ```

Repeat this sequence as many times as necessary.

6.3.2 Importing from Multiple Tapes

To import from multiple tapes, the import tape label must be the same as the one for first export tape. Also, you must have OPER privileges to perform the tasks described in this section.

To direct the output to your terminal rather than the operator's console, issue the REPLY/ENABLE=TAPES command.

To import from multiple tapes, enter the following commands:

```
$ MOUNT/BLOCK=4096 <tape_device_name> <tape_label>
$ IMP <username>/<password>
```

At this point the import starts and you are prompted to enter the import filename as in the following example:

```
Import file:  EXPDAT.DMP > <tape_device_name>:<filename>
```

The import proceeds to the end of the reel.

In the computer room where the tapes are kept, perform the following steps:

1. Make sure the tape drive is allocated.
2. The tape rewinds and dismounts itself.
3. A message flashes onto the operator's terminal instructing to mount the second tape. A request number is provided.
4. The operator mounts the next tape and enters the following statement:

```
$ REPLY/TO=<request_number>
```

Note: Initializing the tape will destroy your export.

Repeat this sequence as many times as necessary.

6.4 Recovering Your Data

If the server is interrupted by a hardware failure, an operating system error, or an unexpected process termination, the result can be damaged files or a database that contains inconsistent data. Recovery is then needed to reconstruct the database in such a way that no committed transactions are lost and no uncommitted changes are retained.

This section describes the procedures for recovering data if media, software, or system fails. You must complete database backups periodically to be able to recover data if you have a media failure.

6.4.1 Overview

Recovering an Oracle9*i* database is the process of restoring normal Oracle9*i* operations when they are interrupted by operating system error, hardware failure, or process termination. Recovery procedures should ensure that no transactions are lost and that no data is written incorrectly. Consequently, you must back up the database regularly.

The first step in recovering normal Oracle9*i* operation is to determine the type of failure that has occurred. There are four types of failure, but only two require action:

- instance failure
- media failure

When either instance or media failure occurs, you need to complete instance or media recovery.

The other two types of failure, statement failure and process failure, result in automatic recovery. For more information about statement and process failure, refer to the *Oracle9i Database Administrator's Guide*.

Instance recovery is done automatically whenever an instance is started. It can be performed after instance failure by shutting down and then restarting the instance. Media recovery is similar to instance recovery, but requires the use of database backups or archived redo logs.

Both instance and media recovery consist of the following two tasks:

- Rolling transactions forward, to redo work that was performed just before the failure
- Rolling transactions backward, to undo work that was performed but not committed before the failure

Refer to the *Oracle9i Database Administrator's Guide* and to the *Oracle9i Database Utilities* for information about the Oracle9*i* utilities used in recovery procedures.

6.4.2 Recovering from Instance Failure

An instance has failed when work executed within the instance has stopped, meaning that read and write transactions are no longer being processed. Instance failure can be caused by loss of power, machine malfunction, an operating system crash, or another hardware or software problem. You can diagnose instance failure by checking if one or more of the detached processes have terminated, or if work in the instance seems to be suspended.

To recover from instance failure, simply restart the failed instance to restore it to the working state that existed immediately before it failed. Whenever an instance is started, the following occurs:

- Both committed and uncommitted transactions recorded in the redo logs are rolled forward.
- Uncommitted transactions are rolled back.
- All locks on Oracle9*i* resources are released.

To restart an instance after it has failed, perform the following steps:

1. Shut down the instance with the command SHUTDOWN. You must use either the IMMEDIATE or ABORT option with the command.
2. Restart the instance with the command STARTUP as normal.

When the instance is restarted, check the trace files generated in the dump directory by the detached processes. Sometimes the failure of one or more of the detached processes will cause instance failure. If possible, the problem that caused process failure should be diagnosed and corrected to avoid recurrence of the problem.

On OpenVMS Clusters where multiple instances reside on different nodes, a failed instance will be recovered by one of the remaining functional instances within the cluster. You must still restart the failed instance, however.

6.4.3 Recovering from Media Failure

A media failure occurs when a nonrecoverable error occurs during a read or write transaction involving one or more of the database files. For example, a disk head crash that causes the loss of any one of the log files, control file, and database files associated with a particular database constitutes media failure. If you are prepared for media failure properly, you can recover both the system tablespace datafiles and the non-system tablespace datafiles.

6.4.3.1 Media Recovery

Media recovery achieves the same results as instance recovery. However, because media failure usually involves loss of data in the database files, media recovery usually requires the use of database backups and archived redo logs. Consequently, you cannot complete a full media recovery automatically as these backups and archived logs are kept offline. Full media recovery requires rather extensive preparation before media failure actually occurs; the following sections describe the actions involved in this preparation.

- Database files for the SYSTEM tablespace
- Database files in other tablespaces
- Online redo logs
- Control files

The procedures for recovering these structures are documented in the *Oracle9i Database Administrator's Guide*.

Note: If you run in RAC mode, you must shut down **all** instances and start up only one instance in exclusive mode to do media recovery.

If you have suffered from media failure, it is unlikely that any of the instances are still operational.

If you need to use an archived redo log file during any of these procedures, use the OpenVMS BACKUP utility to copy the archived file from the archive destination. When prompted to supply the log file sequence number, provide the file specification. Provide the full specification if the location is other than the current device and directory. Wildcards are not accepted.

6.4.3.2 Restoring from an Export File

Refer to *Oracle9i Database Utilities* for information on how to recover from an export file as part of media recovery. If you decide to import from an export file as part of media recovery, you need to recreate the database using the SQL*Plus utility before importing the export file.

1. Back up the current database, redo log, and control files with the OpenVMS BACKUP utility.
2. Create a new database to import into, either manually or by using the Database Configuration Assistant.
3. Create a second rollback segment in the SYSTEM tablespace. Refer to the *Oracle9i Database Administrator's Guide* for more information on creating rollback segments.
4. Import the data from the dump files which were created by the EXP utility.

Note: Private rollback segments can be taken online manually while the database is open using the following SQL command:

```
SQL> ALTER ROLLBACK SEGMENT <name> ONLINE;
```

Part II

Using and Administering the Oracle Tools

Part 2 of this guide consists of the following chapters:

7

Administering SQL*Plus

This chapter describes how to use and administer SQL*Plus on Oracle9*i*.

This chapter contains the following topics:

7.1 Administering SQL*Plus

SQL*Plus commands are described in detail within the *SQL*Plus User's Guide and Reference*. Please refer to that document.

7.1.1 Using Setup Files

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

7.1.1.1 Using the Site Profile File

The global site profile file is `ORA_ROOT:[sqlplus.admin]login.sql`. If a site profile already exists at this location, it is overwritten when you install SQL*Plus.

7.1.1.2 Using the User Profile File

The user profile file is `login.sql`. SQL*Plus looks for this file in the current directory, and then in the directories you specify using the SQLPATH environment variable. Set this environment variable to a comma-separated list of directories. For example:

```
define SQLPATH "Disk1:[Path1],Disk2:[Path2]"
```

SQL*Plus searches these directories for the `login.sql` file in the order they are listed.

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

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

7.1.2 Using the PRODUCT_USER_PROFILE Table

During a typical installation, the PRODUCT_USER_PROFILE table is created automatically. This table is used to disable the SQL and SQL*Plus commands you specify. To recreate this table, run the `ORA_ROOT:[sqlplus.admin]pupbld.sql`script in the SYSTEM schema.

For example, enter:

```
$ sqlplus SYSTEM/MANAGER
SQL> @ora_root:[sqlplus.admin]pupbld.sql
```

7.1.3 Using Demonstration Tables

SQL*Plus is shipped with demonstration tables that you can use for testing.

7.1.3.1 Performing a Typical Installation

During a Typical installation, the user SCOTT and the demonstration tables are created automatically.

7.1.3.2 Creating Demonstration Tables Manually

Use the `ora_sqlplus_demo:demobld.sql` SQL script to create the demonstration tables. In SQL*Plus, you can use any user name to run the `demobld.sql` file to create the demonstration tables in a schema. For example, enter:

```
$ sqlplus system/manager
SQL> @ora_sqlplus_demo:demobld.sql
```

7.1.3.3 Deleting Demonstration Tables

Use the `ora_sqlplus_demo:demodrop.sql` script to drop the demonstration tables. In SQL*Plus, you can use any user name to drop the demonstration tables in the user's schema. For example, enter:

```
$ sqlplus scott/tiger
SQL> @ora_sqlplus_demo:demodrop.sql
```

Note: Both the `demobld.sql` and `demodrop.sql` scripts drop the EMP, DEPT, BONUS, SALGRADE, and DUMMY tables. Before you run the `demobld.sql` script, make sure that these tables do not exist or are not in use for other purposes.

7.1.4 SQL*Plus Online Help

This section describes how to install and remove the SQL*Plus online help.

See Also: For more information on the SQL*Plus online help, refer to *SQL*Plus User's Guide and Reference*.

7.1.5 Installing the SQL*Plus Online Help

There are four ways to install the SQL*Plus online help:

- Perform a typical installation

  When you copy a starter database with pre-built data files as part of the Typical installation, SQL*Plus automatically installs the SQL*Plus online help.

- Use the Database Configuration Assistant

  You can use the Database Configuration Assistant to create help tables when creating a database.

- Install the help facility manually using the `helpbld.sql` SQL script

  You can run the `ORA_ROOT:[sqlplus.admin.help]helpbld.sql` script to manually install the online help. For example, enter:

```
$ set def ora_root:[sqlplus.admin.help]
$ @ helpins.com
```

Note: The `helpbld.sql` SQL*Plus script drops existing online help tables before creating new tables.

7.1.6 Removing the SQL*Plus Online Help

You can also run the `ora_sqlplus:helpdrop.sql` in SQL*Plus to manually drop the online help tables in a schema. For example, enter:

```
$ sqlplus SYSTEM/MANAGER
SQL> @ora_root:[sqlplus.admin.help]helpdrop.sql
```

7.2 Using SQL*Plus

This section describes how to use SQL*Plus on OpenVMS systems.

7.2.1 Using a System Editor from SQL*Plus

If you enter an ED or EDIT command at the SQL*Plus prompt, the system starts an operating system editor, such as `EDT` or TPU, depending upon how the VMS "edit" symbol is defined.

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

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

```
SQL> DEFINE _EDITOR=EDT
```

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

```
SQL> SET EDITFILE test15:[tmp]myfile.sql
```

SQL*Plus does not delete the temporary file.

7.2.2 Running Operating System Commands from SQL*Plus

Using the HOST command or a dollar sign ($) as the first character after the SQL*Plus prompt causes subsequent characters to be passed to a sub-process.

To return to SQL*Plus, enter `logout`.

For example, to enter one command:

```
SQL> HOST SHOW DEFAULT
```

or

```
SQL> $ SHOW DEFAULT
```

To enter multiple operating system commands from SQL*Plus, enter the ***Host*** or ***$*** command, and press return. SQL*Plus returns you to the operating system prompt.

To return to SQL*Plus, enter:

```
$ logout
```

7.2.3 Interrupting SQL*Plus

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

7.2.4 Using the SPOOL Command

SPOOL causes output from all subsequent SQL commands to be captured in a specified file. The default file extension of files generated by the SPOOL command is `.lis`. To change this extension, specify a spool file containing a period (.). For example, enter:

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

7.3 SQL*Plus Restrictions

This section describes SQL*Plus restrictions.

7.3.1 Resizing Windows

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

7.3.2 Return Codes

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

8

Using Oracle Precompilers and the Oracle Call Interface

This chapter describes Oracle Precompilers and the Oracle® Call Interface.

It contains the following topics:

8.1 Overview of Oracle Precompilers

Oracle precompilers are application-development 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 Oracle9*i* or any other ANSI SQL database management system.

> **See Also:** For general information on Oracle precompilers and interface features, see the *Programmers Guide to the Oracle Precompilers*.

8.1.1 Precompiler Configuration Files

System configuration files for the Oracle Precompilers are located in the `ORA_ROOT:[precomp.admin]` directory. Table 8–1 lists the names of the system configuration files for each precompiler. These files are currently empty; their purpose is to specify command line parameters, such as the include directories.

Table 8–1 System Configuration Files for Oracle Precompilers

Product	Configuration File
Pro*C/C++ release 2 (9.2.0.2)	`pcscfg.cfg`
Pro*COBOL release 2 (9.2.0.2)	`pcbcfg.cfg`
Pro*COBOL release 1.8.77.2.0	`pcccob.cfg`
Pro*FORTRAN release 1.8.77.2.0	`pccfor.cfg`
Object Type Translator release 2 (9.2.0.2)	`ottcfg.cfg`
Oracle SQL*Module for Ada release 2 (9.2.0.2)	`pmscfg.cfg`

8.1.2 SQL*Module

SQL*Module is a development tool that facilitates building and managing large applications that access data in an Oracle database. SQL*Module is available for the Ada language.

Additional Information: Refer to the user's guide and README files for the programming language you are using.

8.1.3 Precompiler Executables

Table 8–2 lists products and their corresponding executable names as well as the OpenVMS symbols associated with them. These images are found in the `ORA_ROOT:[bin]` directory.

8.1.3.1 Precompiler README files

Precompiler README files for the various languages are located in `ORA_ROOT:[precomp.doc.<language>]`.

Table 8–2 Executable Names & OpenVMS Symbols for Precompiler Products

Product	Executable	OpenVMS Symbol
Pro*C/C++ release 2 (9.2.0.2)	`proc.exe`	`PROC`
Pro*COBOL release 1.8.77.2.0	`procob18.exe`	`PROCOB18`
Pro*COBOL release 2 (9.2.0.2)	`procob.exe`	`PROCOB`
Pro*FORTRAN release 1.8.77.2.0	`profor.exe`	`PROFOR`
Oracle SQL*Module for Ada release 2 (9.2.0.2)	`modada.exe`	`MODADA`
Object Type Translator release 2 (9.2.0.2)	`ott.com`	`OTT`

8.2 Precompiling

You invoke the precompilers and SQL*Module generator by using the OpenVMS symbols specified in Table 8–2.

8.2.1 Syntax

Use the following syntax to precompile source files:

```
$ <VMS_symbol> INAME=<filename> <option>=<value> ...
```

where:

<VMS_symbol> is the OpenVMS symbol for the precompiler or SQL*Module; and

<filename> is the name of the source file you want to precompile; and

<option> is the precompiling option available for the Oracle Precompilers program. You can supply any number of option-value pairs, separated by a space. And

<value> is the value for the option specified.

8.2.2 Example

```
$ PROFOR INAME=MYFILE HOST=FORTRAN INCLUDE=ORA_PRECOMP
```

The HOST=***<language>*** identifier is optional. For example, the following command is also valid:

```
$ PROFOR INAME=MYFILE INCLUDE=ORA_PRECOMP
```

The INCLUDE option gives the path to the directory that contains the precompiler include files. If not supplied, the include path defaults to the directory in which the include files are distributed.

You can get a list of options and their values (if you have an Oracle instance running) by entering the appropriate symbol name, for example:

```
$ PROFOR
```

The system will display a list of options and their values for Pro*FORTRAN.

8.2.3 Guidelines and Restrictions

The following guidelines and restrictions apply to precompiling.

8.2.4 Using the HP OpenVMS Alpha Debugger

Precompiler programs can be run with the OpenVMS debugger by compiling the program with the /DEBUG qualifier and linking using the D option with the LNPRO<*language*> symbol.

8.2.5 Using Event Flags

If you use OpenVMS event flags in your source code, make sure none of them are numbered 1-18 before compiling the code for use against the Oracle Server. Event flags 1-18 are reserved for the Server.

8.2.6 Migrating Applications Developed with Pro*C Compilers

When migrating applications developed with Pro*C precompilers, each application must have a unique SQLCA and/or ORACA. Oracle Corporation recommends that you insert the following definition in one module to produce a "defining declaration" of the SQLCA structure:

```
#DEFINE SQLCA_STORAGE_CLASS GLOBALDEF
```

Each of the other modules should have the following global reference to product "referencing declarations."

```
#DEFINE SQLCA_STORAGE_CLASS GLOBALREF
```

This line must precede SQLCA.H.

8.3 Compiling

Ensure that the following conditions are met when using the precompilers listed in this section.

8.3.1 Compiler Options Used to Compile Oracle9*i*

Oracle9*i* is compiled with as few deviations from the default C compiler options as possible and with minimal use of pragma statements.

Under the DEC C 6.4a AP compiler on HP OpenVMS Alpha, the compilation options are as follows for most modules:

```
/decc/nostandard/optimize/debug=trace
```

For the RDBMS, the following options are used on OpenVMS:

```
/decc/nostandard/debug=trace/optimize/prefix=all/gran=long/names=as_is
```

If you compile your code with `/debug=trace`, line numbers in your modules will appear, as appropriate, in Oracle9*i* stack trace listings.

8.3.2 Floating Point Formats

Oracle9*i* is compiled with the default floating point format supported by the C compiler. The conversion routines within Oracle9*i* translate operating system-specific floating point numbers into Oracle9*i* internal floating point representation.

Note: Oracle precompiler and OCI programmers should take special note that Oracle9*i* for HP OpenVMS Alpha is compiled to recognize the G_FLOAT floating point format.

8.3.3 Pro*COBOL

You must specify the `/ANSI` option when you compile the Pro*COBOL demo source files.

8.4 Linking

Use the following methods to link object files:

- LNPRO<*language*>.COM

  LNPRO<*language*>.COM is the standard, suggested linking method.

  Use LNPRO<*language*>.COM to link precompiled files, object files, and SQL*Module files.

- LNOCI.COM to link non "C" OCI programs
- LNOCIC.COM to link OCI C programs
- LOUTL.COM

  Use LOUTL.COM under special circumstances when LNPRO<*language*>.COM is not appropriate. If you decide to use LOUTL.COM, use a command syntax similar to that found in the appropriate LNPRO<*language*>.COM script.

- LNPROC == "@ora_proc:lnproc"
- LNPROCOB == "@ora_procob:lnprocob"
- LNPROCXX == "@ora_proc:lnprocxx"
- LNPROFOR == "@ora_profor:lnprofor"
- LNMODADA=="@ora_modada:lnmodada"

Note: All Oracle third party and user tools must link against Oracle with the LOUTL "T" or "Z" option and connect to the database over Oracle Net. Client applications that wish to connect to a database on the same machine should use the bequeath adapter. Applications connecting to remote databases must use the Oracle Net TCP/IP adapter.

8.4.1 Syntax

To link compiled PRO<*language*> object files, use the LNPRO<*language*> symbol. Refer to Table 8–3, " Linking Precompiled Programs" to find the description for each argument.

Table 8–3 Linking Precompiled Programs

Argument	Description
<*language*>	Abbreviation for the programming language you are using (C, COB, FOR).
<*executable*>	Name of the executable image to be created; a filename extension is optional.
<*objectfilelist*>	Comma-separated list of object files and libraries. If this list is longer than one line, use the continuation character,-. Note that there are no spaces in this specification.
<*options*>	List of options with no separators needed: "D" links with the OpenVMS DEBUG utility. "F" produces a full map. "M" creates a link map. "X" produces a link map with cross references.

Example To link MYOBJ and SUB into an COB executable called MYFILE (and to specify options D, and M), use the following command:

```
$ LNPROCOB MYFILE MYOBJ,SUB DM
```

8.4.2 Including Option Files

When you are using the OpenVMS linker, you sometimes give linker directives through standard input. When using the Oracle linking symbols (LNPROC, LNPROADA, etc.), however, you must put your directives into an options file, even if you only have one or two directives.

For example, the following statement is *incorrect*:

```
$ LNPROC MYPROG.EXE MYOBJ,SYS$INPUT/OPT
```

But the following command would work:

```
$ LNPROC MYPROG.EXE MYOBJ,MYOPT/OPT
```

where "MYOPT" is the options file.

8.4.3 Guidelines

When using the link scripts, you should be aware of the following guidelines:

Note: Shared SQLLIB is not supported if you use the "Z" or "T" standalone link options.

8.4.3.1 Using the demos

Several sample programs, covering different aspects of precompiler programs, are provided in the PRO<*language*> demo directories. We recommend that you precompile, compile, and link these programs. You can use these programs as models for new programming efforts.

Before running the Proc Demos, make sure you define the following environment setting:

```
define sys ora_root:[precomp.public.decc]
```

8.4.3.2 Compatibility with ANSI Standard Compilers

Oracle Corporation makes every effort to ensure compatibility with the ANSI standard compilers supported by Hewlett-Packard Computer Corporation. However, new functionality available with the latest compilers might not yet be supported.

8.4.3.3 Linking Shareable Images with LOUTL.COM

You may link a shareable image against Oracle9*i* code using the D option with LOUTL.COM or one of the LNPRO*.COM link scripts that internally calls LOUTL.COM. To eliminate missing file errors, also use the LOUTL flag LS (for Link Shareable).

You may want to install the shareable image in system memory with a command like:

```
$ install create/share/write/header <shareable_image>
```

To avoid receiving an error when you link your main program, including the shareable image in the link list with the option D for debug, also use the LOUTL option LD (for Link Debugger).

In summary, only use the LS option when linking the shareable using the I and D options and use the option LD when linking an executable image that uses this shareable. Or use both the LS and LD options when linking an executable image that uses this shareable.

Note: Make sure that the LS and LD options are separated by spaces from all other options, otherwise they will not be recognized. Also, make sure to specify the S option, which means link shared, and the D option, which means link debug, before the LS and LD options. Other options may occur after LS and LD and may be concatenated.

Valid examples:

DS LS LD
SD LS LD
S LS
D LS
S LD MF

Examples that will fail:

LS S (the S option can not occur after the LS option)
SLD (space is missing between S and LD)
S D LSLD (space missing between LS and LD)
S LDMF (space missing between LD and M)

8.4.3.4 Watching the Link Command Passed to LOUTL

LOUTL looks for the symbol SHOW_LINK_COMMAND, which allows you to see the LINK command that is constructed by LOUTL.COM without waiting for a link map. If this symbol is defined to any non-null value, LOUTL displays the link command. If this symbol is undefined, LOUTL issues the link command silently.

8.4.3.5 Using LNK$LIBRARY When Linking Against Oracle

All Oracle link scripts call LINK with the /NOUSERLIBRARY qualifier. This means that any libraries you want to link automatically using the LNK$LIBRARY logical names will be ignored. Therefore, explicitly include these libraries in your link line or via an option file.

8.5 Using the Oracle Call Interface Routines

Oracle Call Interface routines allow high-level language applications to access data in an Oracle database. Programs that use the OCI routines can make direct calls to Oracle subroutines; they need not be precompiled. C, FORTRAN, and COBOL are supported on HP OpenVMS Alpha for OCI programs.

OCI sample programs are supplied in the following directory:

```
ORA_ROOT:[RDBMS.DEMO]
```

8.5.1 Guidelines

The following guidelines apply to using OCI routines:

- You can run OCI programs with the OpenVMS debugger by compiling with the /DEBUG directive and then linking using the D option of the LNOCIC (or LNOCI) command file (see syntax below).
- While in an asynchronous system trap (AST), you are restricted to using only OBREAK. No other OCI calls can be used.

8.5.2 CDA/LDA Structure Information

For the C OCI programmer, the CDA and LDA structures (64 bytes each) are declared in the header file OCIDFN.H.

The following tabulation of the size and offsets of the structure elements allows COBOL and FORTRAN programmers to use these structures.

8.5.2.1 Alpha Size and Offsets of Structure Elements

The information for the cda_def structure, of which Cda_Def and Lda_Def are typedefs, is shown in Table 8–4:

Table 8–4 Alpha Size and Offsets of Structure Elements

Structure Element	Offset (Bytes)	Size (Bytes)
v2_rc	0	2
ft	2	2
rpc	4	4
peo	8	2
fc	10	1
rc	12	2
wrn	14	1
rid	20	16
ose	36	4
rcsp	44	4

8.5.3 Linking Oracle Call Interface Programs Written in C

LNOCIC.COM is used to link Oracle Call Interface routines written in C. The syntax is:

```
$ LNOCIC <executable> <objfilelist> <options>
```

where:

`<executable>` is the name of the executable image to be created; a filename extension is not required; and

`<objectfilelist>` is a list of object files and libraries separated by commas. If this list is longer than one line, use the continuation character, the hyphen (-). Note that spaces are not allowed in the object file list; and where:

`<options>` is a list of options with no separators needed:

D - Links with the OpenVMS DEBUG utility.

F - Produces a full map.

M - Creates a link map.

X - Produces a link map with cross references.

For example:

```
$ LNOCIC SAMPLE OBJECT1 D
```

8.5.4 Linking Oracle Call Interface Programs Written in Other Languages

LNOCI.COM is used to link with non-C programs. Of these, only FORTRAN, and COBOL are supported on HP OpenVMS Alpha for OCI programs. The syntax is:

```
$ LNOCI <executable> <objectfilelist> <options>
```

For example:

```
$ LNOCI SAMPLE OBJECT1 D
```

Note: The Old Style Oracle Call Interface (HLI) function calls are not supported with the Oracle9*i* Enterprise Edition.

8.5.5 Unexpected Link Errors

If you receive unexpected link errors, you should make a custom copy of the appropriate ORA_RDBMS:LNOCI<*option*>.COM or ORA_PRO<language>:LNPRO<*language*>.COM file, and comment out the following line:

```
$ SHARED_CORE_LIB = "YES"
```

Relink the user-written utility with the custom copy of either ORA_RDBMS:LNOCI<*option*>.COM or ORA_PRO<*language*>:LNPRO<*language*>.COM. If commenting out this line eliminates your link errors, you must run that user-written utility without utilizing shared core.

8.6 Data Areas and Datatypes

Datatypes for Oracle on OpenVMS are described below. Cursor Data Area is correct for HP OpenVMS Alpha as shown in the programmatic interface guides.

8.6.1 Binary Integers

For OpenVMS, binary integers are 32 bits and short binary integers are 16 bits, as shown in Table 8–5.

Table 8–5 Usage of Binary and Short Binary Integers

Programming Language	Usage of Binary Integers	Usage of Short Binary Integers
C	int; short	n/a
FORTRAN	INTEGER*4	n/a
COBOL	PIC S9(9) COMP	PIC S9(4) COMP

8.7 Using Literals as Call Arguments

In FORTRAN, literals and the CHARACTER datatype are passed by descriptor to subroutines. Oracle requires all data to be passed by reference. OpenVMS FORTRAN provides the %REF compiler directive for overriding the normal calling mechanism; %REF should be used to pass literal strings and CHAR data to Oracle.

For example:

```
CALL ORLON (LDA(1), HDA(1), %REF('SCOTT'), 5, %REF('TIGER'), 5)
```

8.8 Optional or Missing Parameters

In HP OpenVMS Alpha, C does not allow missing optional parameters; all call parameters must be specified. FORTRAN and COBOL, however, allow for missing trailing parameters; Oracle provides the necessary defaults. FORTRAN also allows missing embedded parameters; Oracle provides the necessary defaults.

If you omit a parameter using the -1 convention, the argument can be either a reference to the integer -1 or the integer value -1, as long as the argument is of datatype integer or short binary integer (for example, for length specifications). If the argument is the address of any datatype, the -1 must be passed by value.

The following two examples show how to override the normal calling mechanism. In FORTRAN, you could use the following:

```
CALL ORLON(LDA(1), HDA(1), %REF('SCOTT/TIGER'), -1, X, %VAL(-1))
```

In COBOL, you could use this:

```
01 DEFLT PIC S9(9) COMP VALUE -1.
01 LDA PIC X(64).
01 HDA PIC X(256).
01 UID PIC X(11) VALUE 'SCOTT/TIGER'
01 UIDL PIC S9(9) VALUE 11.
CALL ORLON USING LDA, HDA, UID, UIDL,
BY VALUE DEFLT.
```

8.9 Using Event Flags

Event flags signal the completion of synchronous and asynchronous events in HP OpenVMS Alpha, such as disk I/O, terminal I/O, timers, the return of system and user information, lock acquisition, and user interrupts.

Oracle9*i* prevents asynchronous events from interfering with synchronous events by overwriting their event flags. This may increase the reliability of Oracle9*i* software on modern hardware, but it may introduce some problems for application programmers.

Oracle9*i* makes hard-coded references to event flags 1 - 18. All of these event flags except flags 1 and 5 are tied to specific asynchronous events within Oracle9*i*. Event flags 1 and 5 are used by all synchronous events within Oracle9*i* and can also be used by application programmers. SYS$GETEF() is not used for these event flags.

Oracle Net also uses additional event flags, which it gets dynamically from SYS$GETEF() calls from the second event flag group that ranges from 32-63. Make sure that you check the availability of any event flags you use in this range.

Note: Record Management Services (RMS) uses event flags 27 through 31.

A

Messages and Codes

This appendix lists some of the Oracle9*i* messages, codes and actions that are specific to the HP OpenVMS Alpha environment. These messages and codes supplement those in the *Oracle9i Database Error Messages* guide. Refer to that document for a complete list of messages and detailed information about Oracle messages and codes.

This chapter also contains messages and codes that are common in, yet not reserved specifically for, the HP OpenVMS Alpha environment. All messages between 07500 and 07999 are OpenVMS operating system dependent.

%DCL-W-ACTIMAGE: error activating image <*image name*>

Cause: This is an OpenVMS error message that occurs when you try to run an Oracle9*i* tool without installing the Oracle9*i* shareable image.

Action: Install Oracle9*i* in shared memory before the instance is started by executing the following command files:

```
$ INSORACLE
```

ORA-01031:insufficient privileges

Cause: If the correct process rights identifier has not been defined, this error occurs when you try to connect to a database using the CONNECT /AS SYSDBA command.

Action: Set the correct process rights identifier. The following information discusses the process rights identifiers and the privileges needed to control instances.

Privileges to use the CONNECT / depend on:

- whether an ORA_<*sid*>_DBA identifier is in the OpenVMS rights database
- whether the account has the process rights identifier ORA_DBA, ORA_<*sid*>_DBA, or both

These identifiers are added by running the OpenVMS AUTHORIZE utility. The following cases identify process rights identifiers and your subsequent privileges:

- If the identifier, ORA_<*sid_x*>_DBA, exists in the HP OpenVMS Alpha rights database for instance <*sid_x*>, then your account must have been granted the process rights identifier ORA_<*sid_x*>_DBA to control instance <*sid_x*>.
- If the identifier, ORA_<*sid_x*>_DBA, exists in the HP OpenVMS Alpha rights database for instance <*sid_x*>, and your account does not have the process rights identifier ORA_<*sid_x*>_DBA but it does have ORA_DBA, then your account does not have sufficient privileges to control instance <*sid_x*>, but it may control all other instances that do not have ORA_<*sid_x*>_DBA identifiers defined for them.
- If the identifier, ORA_<*sid_x*>_DBA, does not exist in the HP OpenVMS Alpha rights database for instance <*sid_x*> and you have the process rights identifier to ORA_DBA, then your account has sufficient privileges to control instance <*sid_x*> and all other instances that do not have ORA_<*sid*>_DBA identifiers defined for them.

ORA-07515:sfccf: UIC group <= MAXSYSGROUP - file operations not allowed

Cause: File is not created because allowing DBAs to perform file operations if their account's UIC group is less than or equal to the SYSGEN parameter MAXSYSGROUP poses a security risk.

Action: Make sure that the DBA creating or opening database files, redo log files, etc., has a UIC group greater than MAXSYSGROUP.

ORA-07516:sfccf: $open file error

Cause: HP OpenVMS Alpha system service $OPEN failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07517:sfccf: existing file size mismatch with specified file size

Cause: A file that was specified by REUSE already exists but differs in size.

Action: Specify a file size equal to that of the existing file or do not use REUSE.

ORA-07519:sfccf: REUSE not allowed since file owner group <= MAXSYSGROUP

Cause: File is not created because allowing the ORACLE server to REUSE files owned by users with a UIC group less than or equal to the SYSGEN parameter MAXSYSGROUP poses a security risk.

Action: Ensure that no database files, log files, or control files that you attempt to reuse are owned by an account with a UIC group less than or equal to the SYSGEN parameter MAXSYSGROUP. If any valid ORACLE files exist with such ownership conditions, you must change their ownership before attempting to REUSE them.

ORA-07520:sfccf: illegal logical block size

Cause: An illegal logical block size was specified in the parameter file. The block size must be positive, a multiple of 512, and less than the maximum physical I/O data size.

Action: Change DB_BLOCK_SIZE in the parameter file to conform to these limits.

ORA-07521:sfccf: $create file error

Cause: HP OpenVMS Alpha system service $CREATE failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07522:sfccf: new file exists

Cause: A file that was not designated as REUSE already exists.

Action: Add REUSE to the file specification or delete the existing file.

ORA-07526:sfifi: illegal logical block size

Cause: An illegal logical block size was specified in the parameter file. It must be positive, a multiple of 512, and less than the maximum physical I/O data size.

Action: Change DB_BLOCK_SIZE in the parameter file to conform to these limits.

ORA-07527:sfifi: UIC group <= MAXSYSGROUP - file operations not allowed

Cause: File is not created because allowing DBAs to perform file operations if their account's UIC group is less than or equal to the SYSGEN parameter MAXSYSGROUP poses a security risk.

Action: Make sure that the DBA creating or opening database files, redo log files, etc. has a UIC group greater than MAXSYSGROUP.

ORA-07533:sfifi: Cannot open file since file owner group <=MAXSYSGROUP

Cause: File is not created because allowing the ORACLE server to open files owned by users with a UIC group less than or equal to the SYSGEN parameter MAXSYSGROUP poses a security risk.

Action: Ensure that no database files, log files, or control files that you attempt to reuse are owned by an account with a UIC group less than or equal to the SYSGEN parameter MAXSYSGROUP. If any valid ORACLE files exist with such ownership conditions, you must change their ownership before attempting to open them.

ORA-07537:sfccf: Cannot create file since file owner group <= MAXSYSGROUP

Cause: File is not created because allowing the ORACLE server to CREATE or REUSE files owned by users with a UIC group less than or equal to the SYSGEN parameter MAXSYSGROUP poses a security risk.

Action: Ensure that no database files, log files, or control files that you attempt to reuse are owned by an account with a UIC group less than or equal to the

SYSGEN parameter MAXSYSGROUP. If any valid ORACLE files exist with such ownership conditions, you must change their ownership before attempting to REUSE them. Likewise, if you attempt to create a file that will inherit an illegal ownership from the parent directory, you should create it in a different location, or take other steps to avoid this situation.

ORA-07545:sfcmf: $PARSE failure (filename syntax)

Cause: HP OpenVMS Alpha system service failed due to a syntax error when trying to add a new file to the database.

Action: Examine system error and correct filename syntax.

ORA-07546:sfcmf: new file exists

Cause: The filename of a file to be added resolved to that of a file already in the database.

Action: Change the filename of the file to be added.

ORA-07547:sfcmf: $OPEN failure

Cause: HP OpenVMS Alpha system service $OPEN failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07548:sftopn: Maximum number of files already open

Cause: Too many test files open.

Action: This is an internal error. Verify that you can reproduce the error and contact Oracle Support Services.

ORA-07553:sfofi: out of open files

Cause: The number of open files has exceeded an OpenVMS Oracle9*i* compile time limit.

Action: This is an internal error. Verify that you can reproduce the error and contact Oracle Support Services.

ORA-07556:sfotf: $create error

Cause: HP OpenVMS Alpha system service $CREATE failed.

Action: Examine system error message and refer to the OpenVMS system documentation.

ORA-07557:ssfctf: illegal logical block size specified for tape file

Cause: An illegal logical block size was specified for the tape file.

Action: This is an internal error; please contact customer support.

ORA-07558:ssfctf: $create error

Cause: HP OpenVMS Alpha system service $CREATE failed

Action: Examine system error message and refer to the OpenVMS system documentation.

ORA-07560:sltln: $trnlog error

Cause: Translation of a logical name failed (for example, due to overflow, too many levels of logical names, or the logical name was not defined at all).

Action: Define the logical name or look for a name like ORACLE_SID that is exceptionally long or defined circularly. If none, report as a bug.

ORA-07563:sldext: $PARSE failure

Cause: HP OpenVMS Alpha system service $PARSE failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07564:sldext: wildcard in filename or extension

Cause: A wildcard was used in the filename.

Action: Reenter the filename completely.

ORA-07565:sldext: $SEARCH failure

Cause: HP OpenVMS Alpha system service $SEARCH failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07568:slspool: $OPEN failure

Cause: HP OpenVMS Alpha system service $OPEN failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07572:szrfc: insufficient rolename buffer space

Cause: An OS role name was too long.

Action: Redefine the role name to be of correct length.

ORA-07573:slkhst: could not perform host operation

Cause: HP OpenVMS Alpha system service LIB$SPAWN failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07582:spstp: SID has illegal value

Cause: The SID must exist and be less than 6 characters.

Action: Refer to the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha* for information on setting the SID.

ORA-07585:spdcr: $PARSE failure

Cause: HP OpenVMS Alpha system service $PARSE failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07586:spdcr: $SEARCH failure

Cause: HP OpenVMS Alpha system service $SEARCH failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07587:spdcr: $CREPRC failure

Cause: HP OpenVMS Alpha system service $CREPRC failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07620:smscre: illegal database block size

Cause: An illegal database block size was specified in the parameter file. The block size must be positive, a multiple of 512, and less than the maximum physical I/O data size.

Action: Change DB_BLOCK_SIZE in the parameter file to conform to these limits.

ORA-07621:smscre: illegal redo block size

Cause: An illegal redo log buffer size was specified in the parameter file. The buffer size must be positive and a multiple of 512, and less than the maximum physical I/O data size.

Action: Change LOG_BUFFER in the parameter file to conform to these limits.

ORA-07622:smscre: $CREATE failure

Cause: While creating the system global area (SGA) backing file, HP OpenVMS Alpha system service $CREATE failed.

Action: Examine the system error message and refer to the OpenVMS system documentation.

ORA-07623:smscre: $CRMPSC failure

Cause: While creating the system global area (SGA), HP OpenVMS Alpha system service $CRMPSC failed.

Action: Examine the system error message and refer to the OpenVMS system documentation.

The error is caused when there are not enough contiguous global pages available to create the SGA. For example, the SGA created by the distributed INIT.ORA file requires 390000 contiguous global pages. In addition, remember that contiguous global pages are consumed by the installation of the ORACLE shareable image, and any ORACLE tools installed by INSUTILITY.COM.

To show the maximum number of contiguous global pages use the following lexical function:

```
$ WRITE SYS$OUTPUT F$GETSYI("CONTIG_GBLPAGES")
```

To show the number of global pages available use the following lexical function:

```
$ WRITE SYS$OUTPUT F$GETSYI("FREE_GBLPAGES")
```

If the available global pages are fragmented, then reboot the machine after increasing the SYSGEN parameter, GBLPAGES (global page limit). This parameter cannot be dynamically increased. You need to reboot your machine for these changes to take effect. If the available global pages are merely fragmented, but their number is sufficient, rebooting the machine is enough; in that case there is no need to increase the SYSGEN parameter GBLPAGES.

ORA-07625:smsget: $MGBLSC failure

Cause: While mapping the system global area (SGA) during logon, the HP OpenVMS Alpha system service $MGBLSC failed. The usual reason is that Oracle9*i* has not been started up.

Action: Examine the system error message and refer to the OpenVMS system documentation. Start up Oracle9*i* if it is not already started.

ORA-07626:smsget: SGA already mapped

Cause: An attempt to map the SGA during logon failed because it was already mapped. This is an internal error.

Action: Exit your program and try again, and report this to Oracle Support Services.

ORA-07627:smsfre: $CRETVA failure

Cause: While unmapping the system global area (SGA) during logoff, HP OpenVMS Alpha system service $CRETVA failed.

Action: Examine the system error message and refer to the OpenVMS system documentation.

ORA-07628:smsfre: SGA not mapped

Cause: An attempt to unmap the SGA during logoff failed because it was not mapped. This is an internal error.

Action: Exit your program and try again, and report this to Oracle Support Services.

ORA-07636:smsdbp: $MGBLSC failure

Cause: While attempting to set protection in the database buffer debug mechanism, HP OpenVMS Alpha system service $MGBLSC failed.

Action: Verify that you can reproduce the error and contact Oracle Support Services.

ORA-07640:smsget: SGA not yet valid. Initialization in progress

Cause: An attempt was made to map to the SGA while it was being initialized.

Action: Wait until initialization is complete, then try again.

ORA-07647:sszfck: $OPEN failure

Cause: While attempting to reopen a file, HP OpenVMS Alpha service $OPEN failed.

Action: Examine the system message and refer to OpenVMS system documentation.

ORA-07655:slsprom: $TRNLOG failure

Cause: While attempting to translate SYS$INPUT during a prompt for a password, HP OpenVMS Alpha system service $TRNLOG failed.

Action: Examine the system error message and refer to the OpenVMS system documentation.

ORA-07688:smscre: $CREATE_REGION_64 failure

Cause: HP OpenVMS Alpha system service $CREATE_REGION_64 failed.

Action: Examine system error message and refer to OpenVMS system documentation.

ORA-07689:smscre: $CRMPSC_GFILE_64 failure

Cause: HP OpenVMS Alpha system service $CRMPSC_GFILE_64 failed.

Action: Examine system error message and refer to the OpenVMS system documentation.

ORA-07690:smscre: $CRMPSC_GDZRO_64 failure

Cause: HP OpenVMS Alpha system service $CRMPSC_GDZRO_64 failed.

Action: Examine system error message and refer to OpenVMS system documentation.

ORA-07691:smscre: Identifier ORA_SGA does not exist.

Cause: HP OpenVMS Alpha system service: $GRANTID failed.

Action: Add ORA_SGA identifier to the system.

ORA-07692:ssmsget: $MGBSLC_64 failure

Cause: HP OpenVMS Alpha system service $MGBLSC_64 failed.

Action: Examine system error message and refer to the OpenVMS system documentation.

ORA-07693:ssmsget: $DELTVA_64 failure

Cause: HP OpenVMS Alpha system service $DELTVA_64 failed.

Action: Examine system error message and refer to the OpenVMS system documentation.

ORA-07694:ssmsget: $CREATE_REGION_64 failure

Cause: HP OpenVMS Alpha system service $CREATE_REGION_64 failed.

Action: Examine system error message and refer to OpenVMS system documentation.

ORA-07696:smsfre: $DELETE_REGION_64 failure

Cause: HP OpenVMS Alpha system service $DELETE_REGION_64 failed.

Action: Examine system error message and refer to the OpenVMS system documentation.

ORA-07697:smscre: $GRANTID failure

Cause: HP OpenVMS Alpha system service $GRANTID failed.

Action: Examine system error message and refer to the OpenVMS system documentation.

ORA-07698:smsget: $GRANTID failure

Cause: HP OpenVMS Alpha system service $GRANTID failed.

Action: Examine system error message and refer to the OpenVMS system documentation.

ORA-07711:sksatln: mailboxes and null devices illegal for log_archive_dest

Cause: The user specified a mailbox or null device for LOG_ARCHIVE_DEST.

Action: Specify a valid archival device.

ORA-07741:slemop: $OPEN failure

Cause: HP OpenVMS Alpha system service $OPEN failed.

Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA-07822:sspscm: SYS$CREMBX failure

Cause: An error was returned from the SYS$CREMBX function while trying to create the process dump mailbox.

Action: Check the system error message and refer to the OpenVMS system documentation.

9

Oracle Net on HP OpenVMS Alpha

This chapter provides general conceptual information about Oracle Net in the HP OpenVMS Alpha environment.

It contains the following topics:

9.1 Oracle Net Configuration Overview

Oracle Net is a communications software product that allows you to create a data management environment to share information stored in Oracle databases. Oracle Net uses the communications protocols supported by various operating systems to provide a distributed processing and distributed database environment for Oracle. Oracle Net also refers to a set of products or adapters that support industry-standard protocols such as TCP/IP.

An Oracle database management system can be configured in one of the following ways:

a. Centralized Configuration

b. Client-Server Configuration, or

c. Distributed Database Configuration

9.1.1 Centralized Configuration

In a centralized configuration, the Oracle9*i* Server and Oracle tool are located on the same machine. This machine is not necessarily on a network and you can access the application through terminals. If you use a centralized configuration, you may use a simple Oracle Net adapter called the *bequeath adapter*, which requires no Oracle Net configuration. However, if you wish to use Oracle Shared Servers, you must configure Oracle Net even in centralized configurations.

9.1.2 Client-Server Configuration

In a Client-Server configuration, the Oracle9*i* Server resides on a multi-tasking server system, and the client side of the applications resides on another computer, such as a workstation or personal computer. Both the client and server are connected by a physical network and communicate via a network protocol such as TCP/IP. In a Client-Server environment, the Oracle application built with an application development tool makes database requests to the server over the network.

9.1.3 Distributed Database Configuration

In a distributed database configuration, users query separate databases as a single database. The major advantage of a distributed database is that users and applications are not required to know where the data resides. You can query database tables by name, regardless of how the network protocols work together to access the appropriate remote database containing the table. Therefore, Oracle Net

users can communicate and share database information stored in different locations, on different computers, with different operating systems. Distributed databases allow local administration of data and can reduce network traffic if the data that is accessed most often at a location can be stored locally.

Oracle Net allows the client and server to communicate over a variety of media and protocols. A client-server configuration allows DBAs to distribute CPU-intensive user interfaces to low-cost workstations. It also allows application users to be greeted with the graphical user interface (GUI) with which they are most familiar.

9.2 Oracle Net Installations

When installing Oracle Net on OpenVMS, you can choose to install the Oracle Net TCP/IP adapter.

In addition, the OpenVMS Mailbox protocol adapter is installed automatically, as is the bequeath adapter, which allows mailbox connections without a network configuration.

Refer to the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha* for instructions on installing Oracle Net. Also refer to the file ORA_RDBMS:READMEVMS.DOC and ORA_NETCONFIG:README_NETCONFIG.DOC for more installation details.

9.3 Oracle Net and the Transparent Network Substrate (TNS)

This section introduces Oracle Net in general terms and describes the components that make up Oracle Net for HP OpenVMS Alpha Release 2 (9.2.0.2).

9.3.1 Using Oracle Net

Oracle Net connects dissimilar networks together and allows client-server transactions to occur transparently. An end user does not have to know that a network exists, because Oracle Net hides the complexity of machine-level interactions by presenting a layer of interconnectivity to the user through its client-server architecture. This layer is called the Transparent Network Substrate, or TNS.

The OpenVMS computer holds the physical Oracle9*i* database and a client workstation, as well as an Oracle Forms application that needs to access the Oracle9*i* database. The HP OpenVMS Alpha computer is the server and the workstation is the client.

The transaction proceeds as follows:

1. The client requests some data.
2. Oracle Net packages the request and sends it to the TNS.
3. TNS routes the packaged request to the server.
4. Oracle Net on the server side unpackages the request and sends it to Oracle9*i*.
5. Oracle9*i* processes the request and sends the requested data to Oracle Net.
6. Oracle Net packages the data and sends it to TNS.
7. TNS routes the data to the client.
8. Oracle Net on the client side unpackages the data and sends it to the application.

9.4 Oracle Net Architecture

Oracle Net consists of the following components:

- Oracle Net Interface
- Transparent Network Substrate
- Oracle Protocol Adapters

9.4.0.1 Oracle Net Interface

The Oracle Net interface bundles or unbundles messages received from TNS. The Oracle Net interface code resides on all nodes that use Oracle Net. On the client (application program) side, the interface bundles the messages received from the application and passes them to TNS for delivery. On the Oracle9*i* Server side, the interface unbundles the messages received from TNS and passes them to the Oracle9*i* Server.

9.4.0.2 Transparent Network Substrate

TNS allows peer-to-peer connectivity where no machine-level connectivity can occur. It provides a user-transparent layer that enables a heterogeneous network consisting of different protocols to function as a homogeneous network. TNS forms a transparent layer to which different network protocols are connected. It provides a network of applications above the existing networks of computers.

9.4.0.3 Oracle Protocol Adapters

The Oracle Protocol Adapters allow TNS and its services to communicate over existing network *communication protocols*. The Protocol Adapters map the functions of the underlying protocol into the equivalent functions within TNS. This mapping of communication functions allows calls to or from TNS to be nonspecific protocol.

The relationship between TNS (the network NT layer), TCP/IP (the network NT layer - in this case, /NTT adapter), and the OpenVMS TCP/IP layer, is as follows: the TNS and the Oracle Protocol Adapters interface with existing network protocols. For any TNS client running an industry-standard protocol, the Oracle Protocol Adapter interfaces between the unique API of the underlying protocol and the consistent interface of Oracle's TNS.

A single machine can support multiple protocols and protocol adapters simultaneously. A node that supports multiple protocols and protocol adapters is said to be a member of multiple TNS communities, one for each protocol installed.

A TNS client belonging to multiple communities is common in two cases:

- As a client application that needs to access other applications in more than one network. Installing two protocols and protocol adapters allows a client to connect to any server application in either community.
- As a server application that is being accessed by clients from multiple TNS communities. Installing two protocols and the protocol adapters allow all clients from both communities to access a server application on that machine.

For more information about Oracle Net, refer to the following manuals:

- *Oracle Net Administrator's Guide*
- *Oracle Advanced Security Administrator's Guide*
- *Oracle Internet Directory Administrator's Guide*

9.5 The Protocol Adapters

This section gives information about the following protocol adapters on HP OpenVMS Alpha:

- IPC - Mailbox Protocol
- TCP - TCP/IP Protocol
- BEQ - Bequeath Protocol
- Bequeath Listener

Note: This section is a supplement to the *Oracle Net Administrator's Guide* guide.

9.5.1 IPC - Mailbox Protocol

The Mailbox protocol adapter, or IPC adapter, is automatically configured for use when you install Oracle Net. It can be used for Client-Server connections when both client and server are on the same OpenVMS node. If the client and server are on different machines, then the connection must take place using TCP/IP.

When configuring the TNS listener to listen for mailbox connections, you need to specify a KEY value in LISTENER.ORA for the IPC protocol. The listener then creates a mailbox which listens for connections and creates a system-wide logical name (the same as the KEY value) which translates to this mailbox device. It is via this logical name that clients find the listener's mailbox.

9.5.1.1 Syntax

The following fields must be defined:

```
(PROTOCOL=IPC)
(KEY=<IPC logical name>)
```

where:

PROTOCOL is the keyword that identifies the specific protocol adapter used; for this protocol, the value is IPC. The value can be entered in either uppercase or lowercase, and

KEY is the logical name used to connect to the listener via the Mailbox adapter.

9.5.1.2 Example

This example shows the two fields for the HP OpenVMS Alpha Mailbox adapter.

```
(PROTOCOL=IPC)
(KEY=ORA_IPC)
```

9.5.2 TCP - TCP/IP Protocol

The TCP/IP protocol adapter provides support for Client-Server connections using TCP/IP as a protocol. You can turn Oracle Net support for TCP/IP on or off via the NetConfig configuration screen (refer to the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha* for more information).

Oracle Net on OpenVMS is developed and certified using Hewlett-Packard's TCP/IP Services for OpenVMS (UCX). If you wish to use the TCP/IP protocol adapter for Oracle Net, you should have Version 5.1 ECO 4 TCP/IP Services for HP OpenVMS Alpha installed. TCP/IP protocol stacks from other vendors may work with Oracle, but customers use these products at their own risk. Any TCP/IP problems that can not be reproduced using TCP/IP Services for HP OpenVMS Alpha will be referred to the TCP/IP vendor.

9.5.2.1 Syntax

The following fields must be defined:

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

The following field is optional:

```
(QUEUESIZE=n)
```

Following the syntax above:

PROTOCOL is the keyword that identifies the specific protocol adapter used; for this protocol, the value is TCP. The value can be entered in either uppercase or lowercase, and

HOST is the host name or IP address, and

PORT# is the TCP/IP port number.

9.5.2.2 QUEUESIZE

Parameter to increase the queue size. This parameter is optional; if it is not specified, the default value is 20. If simultaneous connections are made to the listener, some connection requests may not be received if the listener socket queue size is too small.

Example In this example, the TCP/IP connect descriptor specifies a listener on the ALPHA1 host.

```
(PROTOCOL=TCP)
(HOST=ALPHA1)
(PORT=1526)
```

9.5.3 BEQ - Bequeath Protocol

Each database that you wish to connect with the bequeath protocol adapter must have a command file named ORASRV_BEQ_<***sid***>.COM in ORA_ROOT:[NETWORK.ADMIN]. For databases created with the Oracle7 RDBMS Release 7.3.2 or later, this command file is generated when you create the database. You must create this command procedure manually for pre-existing databases.

Execute the command procedure ORA_NETWORK:CREATE_ORASRV_BEQ.COM as follows:

```
$ @ORA_NETWORK:CREATE_ORASRV_BEQ <ora_db> <sid> <dbname>
```

where:

<ora_db> is the database administration directory;

<sid> is the SID of the database, and

<dbname> is the NAME of the database.

For example:

```
$ @ORA_NETWORK:CREATE_ORASRV_BEQ DKA400:[<home.ORADATA.<db_name>] -  PROD PRODDB
```

9.5.4 Bequeath Listener

On OpenVMS, the Bequeath Listener is used as a default to provide dedicated server connections for a local client. The Bequeath Listener, running as a detached process, creates detached server processes to service clients on the same machine, using the bequeath adapter. This allows the Oracle server to run in a suitably privileged process. The alternative would be to have the server installed with privileges and run in a subprocess of the client. However, that would require the server to be linked without traceback information, making server trace information unusable if problems are encountered.

For each request from the client, the Bequeath Listener creates a detached server process and two mailboxes. It then sends the mailbox names to the client and the client establishes a connection to the server using these mailboxes.

By default, these mailboxes are created with a buffer quota of 8192 bytes and a maximum message size of 2048 bytes. You can change these parameters by defining logical names in the file ORASRV_BEQ_<SID>.com with other values. For example:

```
$ define ORA_BEQ_MBXSIZ n
$ define ORA_BEQ_MBXBFQ n
```

The maximum value for the mailbox buffer quotas is 60000 bytes. You should adjust these values carefully, and you should adjust them for performance reasons only.

The Bequeath Listener uses a known mailbox name to listen for client requests. This mailbox name is in the format:

```
ORA_BEQ_READ_MBX_xxxxxxxxxx_n
```

where:

xxxxxxxxxx is the Oracle image ID unique to the system (padded with zeroes).

n is a single-digit number (0-9) that is the Bequeath Listener number.

9.5.4.1 Starting up the Bequeath Listener

The Bequeath Listener starts automatically when INSORACLE is invoked (at installation time or later, usually during system startup). Unless you decide to invoke the REMORACLE command, the Bequeath Listener should be up and running all the time.

If the Bequeath Listener is down and you want to start it, execute the command:

```
BEQLSNR START
```

9.5.4.2 Bequeath Listener Status

You can issue a status command to determine whether the Bequeath Listener is up and running. Issue the command:

```
BEQLSNR STATUS [n]
```

If you do not provide the optional numeric parameter, then Bequeath Listener 0 is queried. To query Bequeath Listeners 1 through 9, if they exist, supply the number on the command line.

9.5.4.3 Shutting Down the Bequeath Listener

To stop the Bequeath Listener issue the command:

```
BEQLSNR STOP [n]
```

If you do not provide the optional numeric parameter, then all Bequeath Listeners for the installation are stopped. To stop a particular Bequeath Listener, provide its number in the command line.

9.5.4.4 Problem Resolution

This section details the things you can do to resolve problems with Bequeath Listener.

9.5.4.4.1 Writing trace information

The Bequeath Listener writes some trace information, but because the output of the detached processes is set to the null device (NL:), normally you will not see it.

To get the trace information from the Bequeath Listener, perform the following tasks:

1. Stop the Bequeath Listener.
2. Edit the STARTUP_BEQLSNR.COM.
3. Change the NL: to a file name.
4. Restart the Bequeath Listener.

9.5.4.4.2 Changing the quota for a Server Process that is created by the Bequeath Listener

To change the quota, modify the file BEQLSNR.COM and remove the comments for the quota parameter that you want to change. Be sure to STOP/START the Bequeath Listener after modifying this file.

9.5.4.4.3 For all ORA-12203 Problems

Be sure that the image identifier string is present in the ORA_BEQ_READ_MBX system logical name. It must be the same as the equivalence - name for the ORA_BEQ process logical.

To verify this, issue the command:

```
$ show logical *beq*
```

The results displayed will look similar to the following:

```
(LNM$PROCESS_TABLE)
          "ORA_BEQ" = "Z910000000"
(LNM$SYSTEM_TABLE)
          "ORA_BEQ_READ_MBX_Z910000000_0" = "MBA6839:"
```

9.5.4.4.4 Problem: ORA-12203: TNS:unable to connect to destination

If you experience this problem, issue the command BEQLSNR STATUS to determine whether the Bequeath Listener is up and running. If the Bequeath Listener does not respond, use the command BEQLSNR STOP to stop the Bequeath Listener and use the command BEQLSNR START to restart it.

9.5.4.4.5 Client Problem: ORA-12203: TNS:unable to connect to destination

Choose one of the following solutions:

- Change the logical ORA_BEQ_TIMEOUT to something greater than 120 seconds (for example: 300 seconds). Before running the client program, define this logical also in the ORA_NETWORK:BEQLSNR.COM file.

  or

- Define the logical ORA_BEQ_NUM_OF_LISTENERS to a value between 1 and 10 to increase the capacity, when a number of clients are connecting at the same time to the Bequeath Listener.

With this method, you can increase the number of connections that the Bequeath Listeners can handle at one time. Each time that a client requests a connection, it will randomly pick one of the Bequeath Listeners that are running to serve it with the connection request. Note that you do not need to STOP/START the Bequeath Listener after defining this logical name. This logical name determines the number of Bequeath Listeners. However, you do need to start it through ***x***, where ***x*** is the value of that logical.

9.5.4.5 Bequeath Listener Privileges

The Bequeath Listener must have the OpenVMS privileges listed in Table 9–1 to be able to perform the associated functions listed in the table.

Note: Before attempting to start the Bequeath Listener or the TNS Listener, the process that starts the Bequeath Listener must have the privileges in Table 9–1 or be able to have them set. Refer to"TNS Listener Privileges" on page 9-14 for more information about setting TNS Listener privileges.

Table 9–1 Bequeath Listener and TNS Listener Privileges and Their Functions

Privilege	Function
CMKRNL	Pass this privilege to server processes that the Listener creates.
DETACH	Create detached processes.
LOG_IO	Perform certain I/O functions.
PRMMBX	Create a permanent mailbox on which to listen. (The mailbox is permanent so that the logical name associated with it goes into the SYSTEM logical name table.)
SYSLCK	Lock system wide resources.
SYSNAM	Create SYSTEM logical names and shared logical name tables.
SHARE	May assign channels to non-shared devices.
TMPMBX	Create temporary mailboxes.
WORLD	Allow the Listener to get information about and to control processes that it may not have created, such as dispatchers and shared server processes.

9.6 TNS Listener

The function of the TNS Listener is to receive connection requests from local or remote clients and to provide the client with a Server process to which to connect. The Listener can service multiple instances. For each instance, the Listener keeps a list of services that provide access to that instance. If multi-threaded servers are being used, the Listener may direct a client connection to a dispatcher. Otherwise, for dedicated servers, the Listener will direct the client connection to an existing Oracle Shared Server or will create a new server process to service the connection.

In Oracle9*i*, there is a major change in the way the listener is configured for Oracle Shared Servers. The Oracle Shared Server parameters are not the same as in Oracle8*i*. When you configure for Oracle Shared Server, a request for a "Dedicated Server" is no longer handled using the parameters from the `LISTENER.ORA` file. This now happens as part of the dispatcher registration. The PMON process registers dispatchers with the listener for Oracle Shared Server connections. The `SID_LIST_<listener>` section is no longer used to establish dedicated server connections. These are now automatically handled by the listener, which directly uses the script `ORA_ROOT:[NETWORK.ADMIN]ORASRV_NETV2_<SID>.COM` to launch the dedicated server process. This script is automatically created when a new database/instance is created.

General information about the TNS Listener and its configuration can be found in the generic Oracle Net documentation. This section provides only information about the TNS Listener that is specific to HP OpenVMS Alpha. The sections are as follows:

- LSNRCTL
- TNS Listener Privileges
- Process Quotas
- ORASRV_NETV2_<SID> Command File
- General Connections

9.6.1 LSNRCTL

The LSNRCTL utility is used to start and stop the TNS Listener and to query its status or services. The LSNRCTL command executes the command procedure `ORA_NETCONFIG:LSNRCTL.COM`, which provides a shell to the executable program `ORA_ROOT:[BIN]LSNRCTL.EXE`.

The main function of the command procedure is to check that the privileges required to start the TNS Listener are present (as detailed in the next section "TNS Listener Privileges"). If a `LSNRCTL START` command is entered and the required privileges are not present, an error is displayed and `LSNRCTL` exits.

Note: Start the TNS Listener using the Oracle Account.

Caution: If you enter the LSNRCTL interactive mode by giving the LSNRCTL command without a subcommand, and you have received a warning about inadequate privileges, do not attempt to start the Listener. Although the Listener process may start, (depending upon the privileges you have), it may not function properly.

Additional Caution: Also, do not start the Listener from a process that has a UIC in the system group -- for example, a group less than or equal to MAXSYSGROUP. If you enter a LSNRCTL START command from such a process, an error is displayed and LSNRCTL exits. If you enter a LSNRCTL command with no arguments, you are warned not to start the Listener from within the LSNRCTL utility. If the Listener is running in a system group, any Server processes it creates will be in the system group. The Server is aborted, because it does not allow itself to run in privileged groups. On OpenVMS, the Oracle Shared Server must be configured to use only TCP/IP protocol.

9.6.2 TNS Listener Privileges

The process in which the TNS Listener runs must have the OpenVMS privileges listed in Table 9–1 to be able to perform the associated function.

Note: Before attempting to start the TNS Listener, the process that starts the Listener must have the privileges in Table 9–1 or be able to have them set. As noted above, the LSNRCTL command file will attempt to set these privileges and warn the user if it was unable to do so.

9.6.3 Process Quotas

Process quotas for the TNS Listener and for the Server processes which the TNS Listener creates can be controlled by logical names. The logical names are:

ORA_LSNR_*<quotaname>*

where:

<quotaname> can be either or one of these, ASTLM, BIOLM, BYTLM, CPULM, DIOLM, ENQLM, FILLM, JTQUOTA, PGFLQUOTA, PRCLM, TQELM, WSQUOTA, WSDEFAULT or WSEXTENT.

Several of the logical names are defined in LSNRCTL.COM and control the quotas of the TNS Listener process. They are defined in user mode so that they are not present after exiting LSNRCTL. If your TNS Listener supports an especially large number of services, some of these quotas may need to be increased. For the quotas you determine to be deficient or under direction of Oracle Support, you can edit the quota values in LSNRCTL.COM.

To control the quotas of the processes that the TNS Listener creates, specify the logical names in the file ORA_NETWORK:TNSLSNR.COM, the command file that runs in the TNS Listener process. Statements to define these logical names are in TNSLSNR.COM, but are commented out.

If, for example, a very large file backed SGA requires that Server processes have larger quotas, you can uncomment the appropriate logical name definition in TNSLSNR.COM and specify the quota value.

Quotas can also be specified for the Server processes in the LISTENER.ORA file on a SID-by-SID basis. This is done in the SID_DESC section for a TNS Listener. For example:

```
SID_LIST_LISTENER =
    (SID_DESC =
      (SID_NAME = <name>)
      (PROGRAM = <disk:>[<directory>]ORASRV_NETV2_<SID>.COM)
      (OSDS=
        (PRIORITY=<number>)
        (QUOTA=
          (ASTLM=<number>)
          (BYTLM=<number>)
          (PGFLQUOTA=<number>)
        )
      )
    )
```

There are no restrictions on the number of quotas that you can specify in the QUOTA list. However, if any quota is specified in the QUOTA list, then none of the quotas specified by logical name will be used and quotas that are not specified in the list will assume the system default.

Note: The process priority of the Server can also be specified, but this is not recommended.

9.6.4 ORASRV_NETV2_<SID> Command File

The file ORASRV_NETV2_<SID>.COM is automatically created for each SID during creation of the new database/instance.

In an non-Oracle Shared Server situation, the behavior is the same as seen in earlier releases. The "PROGRAM=" parameter should point to this script in the `LISTENER.ORA`. Here's an example:

```
(SID_LIST_LISTENER =
 (SID_DESC =
    (SID_NAME = PROD)
  )
)
```

When the TNS Listener starts a dedicated server process, it extracts the PROGRAM= parameter from the `LISTENER.ORA` file to identify which command procedure to run in the dedicated process.

In an Oracle Shared Server configuration, the TNS Listener need not contain the above-mentioned `SID_LIST_<listener>` section. The Oracle Shared Server dispatchers registers with the TNS Listener directly, including specifying the command procedure to run for a "Dedicated Procedure".

This command procedure is currently hard-coded to be `ORA_ROOT:[NETWORK.ADMIN]ORASRV_NETV2_<SID>.COM` which is created automatically. The location and name-syntax of this file cannot be changed. Even if a `SID_LIST` section is specified in `listener.ora` that might point to the same or different script, it is completely ignored.

9.6.5 General Connections

Make sure that your Oracle Net task file defines any logical names used by the INIT.ORA parameters USER_DUMP_DEST and BACKGROUND_DUMP_DEST (if defined).

9.7 Oracle Names

The function of the Names Server is to resolve connection addresses in a homogeneous and centralized location. As a client issues a connection request, the Names Server is responsible for directing the client connection request to the appropriate TNS Listener for the specified SID. TNSNAMES.ORA can also resolve the listener address. However, the benefits of the centralized list of connection addresses that Oracle Names provides greatly eases the maintenance of large network definitions.

This section provides information about Oracle Names on OpenVMS. It covers the following topics:

- NAMESCTL
- Names Server Privileges

Note: This section assumes that Oracle Names and all related Oracle Net products have been installed at your site.

9.7.1 NAMESCTL

The NAMESCTL utility is used to start and stop the Names Server and to query its status or services. The NAMESCTL command executes the command procedure ORA_NETCONFIG:NAMESCTL.COM, which provides a shell to the executable program ORA_NETCONFIG:NAMESCTL.EXE.

The main function of the command procedure ORA_NETCONFIG:NAMESCTL.COM is to check that the privileges required to start the Names Server are present (refer to the section "Names Server Privileges" for more information). If a NAMESCTL START command is entered and the required privileges are not present, an error is displayed and NAMESCTL exits.

Note: Start the Names Server using the Oracle account.

Caution: If you enter the NAMESCTL interactive mode by giving the NAMESCTL command without a subcommand and you have received a warning about inadequate privileges, do not attempt to start the Names Server. The Names Server process may start (depending upon the privileges you have), but it may not function properly.

Additional Caution: Do not start the Names Server from a process that has a UIC in the system group -- for example, a group less than or equal to MAXSYSGROUP. If you give a NAMESCTL START command from such a process, an error is displayed and NAMESCTL exits. If you enter a NAMESCTL command with no arguments, you are warned not to start the Names Server from within the NAMESCTL utility

9.7.2 Names Server Privileges

The process in which the Names Server runs must have the OpenVMS privileges in Table 9–2 to be able to perform the associated function.

Note: Before attempting to start the Names Server, the process that starts the Names Server must have the privileges in Table 9–2 or be able to have them set. As noted above, the NAMESCTL command file will attempt to set these privileges and warn the user if it was unable to do so.

Table 9–2 Names Server Privileges and Their Functions

Privilege	Function
CMKRNL	Facilitate kernel mode processing.
PRMMBX	Create a permanent mailbox on which to listen (The mailbox is permanent so that the logical name associated with it goes into the SYSTEM logical name table.)
SYSNAM	Create SYSTEM logical names and shared logical name tables.
TMPMBX	Create temporary mailboxes.

9.8 Oracle Intelligent Agent

The Oracle Intelligent Agent is a backend server process that communicates with the Oracle Enterprise Manager (OEM) running on a Windows or UNIX machine.

This section provides information about installing and running the Oracle Intelligent Agent on HP OpenVMS Alpha. Read this section carefully and completely before beginning to install and use the Oracle Intelligent Agent on HP OpenVMS Alpha.

This section covers the following topics:

- Installing the Oracle Intelligent Agent
- Oracle Intelligent Agent Setup and Discovery Option
- Oracle Intelligent Agent Startup, Shutdown, and Status Query
- Oracle Intelligent Agent Maintenance

9.8.1 Installing the Oracle Intelligent Agent

The Oracle Intelligent Agent requires that a supported TCP/IP implementation be installed on your OpenVMS system. In addition, you must enable TCP/IP support for Oracle Net in the NetConfig configuration screen.

For more information, refer to "TNS Listener" on page 9-12 in this chapter.

The Oracle Intelligent Agent may be installed at the same time as other products or it may be installed later.

Installation of the Oracle Intelligent Agent creates the directory `ORA_ROOT:[oemagent]` as an installation directory. It also creates a directory structure under the network subdirectory `ORA_ROOT:[network.agent...]` where most of the Oracle Intelligent Agent files will reside.

If you are using the same Oracle Installation from more than one node in an OpenVMS cluster, you can only run the Oracle Intelligent Agent from this installation on one of the nodes. If you attempt to run the Oracle Intelligent Agent on multiple nodes from the same installation, there will be file name and file usage conflicts. This is a generic limitation of the Oracle Intelligent Agent on clusters for all platforms.

For each additional node on which you wish to run the Oracle Intelligent Agent, you must perform a client-only installation for the Oracle Intelligent Agent (installing AGENT, NETCONFIG, and UTIL) and run the Oracle Intelligent Agent from this client-only installation.

9.8.2 Oracle Intelligent Agent Setup and Discovery Option

To correctly set up the Oracle Intelligent Agent environment, the following two kinds of files need to be created:

- Startup command procedures
- Parameter (*.ORA) files

9.8.2.1 Creating the Startup Scripts

Once the Oracle Intelligent Agent has been successfully installed, create the following two files:

- ORA_ROOT:[NETWORK.AGENT]AGENT_START.COM

To create this file, modify AGENT_START_COM.SAMPLE to replace the definition of symbol ORA_ROOT with the correct path for your installation and save it appropriately as a .COM file. When you startup the Oracle Intelligent Agent, AGENT_START.COM is run as a Detached Process.

9.8.2.2 Oracle Intelligent Agent Parameter Files and Discovery Option

At startup, and when requested by the OEM console thereafter, the Oracle Intelligent Agent runs a `tcl` script called `NMICONF.TCL`, which resides in the ORA_ROOT:[NETWORK.AGENT.CONFIG] directory. This script starts by reading LISTENER.ORA and TNSNAMES.ORA to discover any locally installed Oracle databases and instances. Then it reads an optional ORATAB.ORA file, if present, in the TNS_ADMIN directory.

The ORATAB.ORA file should be as follows:

<sid_name>*,*<network admin directory>

For example:

```
ORA9,disk$d1:[Oracle9.NETWORK.ADMIN]
```

Note that you can specify any SID that you want the Oracle Intelligent Agent to monitor that exists on this node.

Then, for each database instance found in ORATAB.ORA, the tnsnames list is searched for an address on the local host with the appropriate SID in the CONNECT_DATA. The key corresponding to the first matching address in the list becomes the name of the database. The listener.ora found in those same directories is searched for the SID of the database. Again, the first TNS Listener that matches our SID becomes the listener active for that database.

Note: This generic discovery phase is impossible if the local database names are stored in Oracle Names instead of in a local TNSNAMES.ORA file. You cannot do the backwards SID-to-name matching through Names. As a result, if Oracle Names is in use for the host, an old-style SNMP.ORA must still be in TNS_ADMIN, with the parameter dbsnmp.register_with_names set to FALSE. If this flag is detected at startup, none of the generic discovery occurs. Instead, the information in the old-style snmp.ora is used to construct the new configuration files.

The configuration files SNMP_RO.ORA and SNMP_RW.ORA are created and should reside at TNS_ADMIN, the same location as the TNS config files.

The Tcl script NMICONF.TCL can execute other Tcl scripts written specifically to discover other Oracle services. If these other scripts exist, they should be installed with NMICONF.TCL in ORA_ROOT:[NETWORK.AGENT.CONFIG].

The file ORA_OEMAGENT:SERVICES.ORA is created during the discovery phase, and will be used to tell the OEM which services the Oracle Intelligent Agent is monitoring.

9.8.2.3 Setting the Preferred Credentials

The preferred credentials are supported from the OEM console. To run a job on the HOST database, you must supply username/password in the preferred credentials fields in the OEM console. To check that the username/password is valid, login to the HOST node where the Oracle Intelligent Agent is running and issue the following command, to verify that the account is not disabled and that it has the ORA_AGENT_ID identifier:

```
$ show process/right
```

9.8.3 Oracle Intelligent Agent Startup, Shutdown, and Status Query

This section explains how to startup, shutdown, and status query the Oracle Intelligent Agent.

9.8.3.1 Startup of the Oracle Intelligent Agent

The Oracle Intelligent Agent consists of a single process DBSNMP. In addition, when a "job" is executed, it creates a new detached JOB process running DBSNMPJ.COM.

Use the following command to startup the Oracle Intelligent Agent:

```
$ AGENTCTL startup
```

This command creates a detached process with a process name of ORA_AGENTWORK.

If a nonzero trace level is specified, then the Oracle Intelligent Agent creates a trace file with the name `DBSMP_<pid>.TRC` in the `ORA_ROOT:[NETWORK.TRACE]` directory.

Note: The process that starts the Oracle Intelligent Agent must have the GROUP and GRPNAM privileges.

9.8.3.2 Shutdown of the Oracle Intelligent Agent

Use the following command to shut down the Oracle Intelligent Agent:

```
$  AGENTCTL STOP
```

Note: Use the Oracle9*i* account to stop or start the Oracle Intelligent Agent.

Use the following command to verify whether the Oracle Intelligent Agent is running:

```
$ AGENTCTL STATUS
```

9.8.4 Oracle Intelligent Agent Maintenance

Unlike the TNS Listener process, the Oracle Intelligent Agent processes are in a continuous loop, polling for incoming connections in each loop. This means that trace information is continuously being generated. Therefore, it is advisable to turn off tracing during normal operation and to turn it on only when a problem is encountered.

9.9 Advanced Security Option

This section provides OpenVMS-specific installation information for the current release of Advanced Security Option (ASO) for Security and Single Sign-On.

Note: A separate license is required to use ASO.

This section covers the following topics:

- Usage Notes for the Authentication Adapters

9.9.0.1 Task 3: Manual Steps for the Authentication Adapters

In the database server's local INIT.ORA file, set the following parameters:

```
remote_os_authent = false
os_authent_prefix = ""
```

9.9.0.1.1 For Kerberos5 Adapter

The following file is required on the client side:

- KRB.CONF - configuration file that specifies the default realm of the client and maps all known realms to Key Distribution Centers (KDCs).

The following files are required on the server side:

- KRB.REALMS - maps hostnames and domains into realms
- V5SRVTAB - contains key that the KDC uses to encrypt a service ticket for the client

The location of all of the above files must be specified using corresponding parameters in SQLNET.ORA.

Additionally, the Oracle Net client also creates a credential cache file whose location needs to be specified in SQLNET.ORA on the client side.

The following is an example of the parameters in SQLNET.ORA for an installation that can act as both client and server:

```
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=ORACLE
SQLNET.AUTHENTICATION_SERVICES = (BEQ,KERBEROS5)
SQLNET.KERBEROS5_KEYTAB = DISK:[TST901.NETWORK.ETC]V5SRVTAB.
SQLNET.KERBEROS5_CONF = DISK:[TST901.NETWORK.KRB5]KRB.CONF
SQLNET.KERBEROS5_REALMS = DISK:[TST901.NETWORK.KRB5]KRB.REALMS
SQLNET.KERBEROS5_CC_NAME = DISK:[TST901.NETWORK.CCACHE]CCFILE.DAT
```

9.9.1 Usage Notes for the Authentication Adapters

The usage notes are categorized into the following areas:

- General Information
- Kerberos5

9.9.1.1 General Information

Include the following line in your LISTENER.ORA file:

```
SQLNET.AUTHENTICATION_SERVICES=(NONE)
```

The listener should not participate in the authentication service.

It is recommended that you always include BEQ as one of the authentication services in your SQLNET.ORA file. Here is an example:

```
SQLNET.AUTHENTICATION_SERVICES=(BEQ,KERBEROS5)
```

In this way, connections within the server machine through the default bequeath adapter do not have to go through the authentication. This is especially important during database startups and shutdowns.

9.9.1.2 Kerberos5

1. Make sure that the clock skew between the client machine and the machine running the KDC is less than one minute.
2. Oracle client and server processes use the Coordinated Universal Time (UTC) format (time elapsed since 00:00:00 Jan. 1, 1970 in records). Make sure that your system is set to the correct time zone in terms of deviation from Greenwich Mean Time (GMT). Otherwise you will get the error "Clock skew too great" in your Oracle Net trace file.
3. Make sure that the value of the parameter SQLNET.AUTHENTICATION_KERBEROS5_SERVICE that you specify in SQLNET.ORA matches exactly, including case, with the value specified in the KDC.

10

Configuring Oracle Net Services

This chapter decribes the Oracle Net Services features for OpenVMS.

It contains the following topics:

10.1 Core Oracle Net Services Products and Features

This section describes core Oracle Net Services products and features.

> **See Also:** For more information about Oracle networking, refer to the *Oracle9i Net Services Administrator's Guide*.

10.1.1 Oracle Net Services and Utilities

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

10.1.1.1 Location of Oracle Net Services Configuration Files

The default directory for Oracle Net Services configuration files on OpenVMS systems is `ORA_ROOT:[network.admin]` or `TNS_admin`.

Oracle Net Services searches the following locations for configuration files in the following order:

1. For `sqlnet.ora` file, the current working directory from where an application is run.
2. The directory specified by the TNS_ADMIN logical, if set.

For each system-level configuration file, users may have a corresponding local private configuration file (stored in the user's home directory). The settings in the local file override the settings in the system-level file. Table 10–1 lists the system-level configuration files and the corresponding local configuration files:

Table 10–1 Oracle Net Configuration Files

System-Level Configuration Files	Local Configuration Files
`sqlnet.ora`	`tns_admin:sqlnet.ora`
`tnsnames.ora`	`tns_admin:tnsnames.ora`

10.2 Oracle Net Services Protocol Support

Oracle Net Services Release 2 (9.2.0.2) on OpenVMS supports the following protocols:

- Bequeath (BEQ)
- IPC
- TCP/IP

Before installing the TCP/IP protocol support, you must install and configure the appropriate operating system software. The BEQ and IPC protocol supports do not have any specific operating system requirements.

> **See Also:** the *Oracle9i Installation Guide Release 2 (9.2.0.2) for hp OpenVMS Alpha* for more information about Oracle Net Services protocol support.

10.2.1 ADDRESS Specification

The IPC, TCP/IP and Oracle Net Services protocol supports each have a protocol-specific ADDRESS specification that is used for Oracle Net Services configuration files and for the DISPATCHERS initialization parameter in the `init`***`sid`***`.ora` file. Refer to the ADDRESS specification heading under each protocol section in this chapter for more information on protocol-specific ADDRESS specification.

Table 10–2 shows the ADDRESS specifications for each supported protocol.

Table 10–2 ADDRESS Specification Summary

Supported Protocol	ADDRESS Specification

BEQ	`(ADDRESS =`
	
	`)`

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

Table 10–2 (Cont.) ADDRESS Specification Summary

Supported Protocol	ADDRESS Specification
TCP/IP	(ADDRESS = (PROTOCOL=TCP) (HOST=hostname) (PORT=port))

10.3 Bequeath (BEQ) Protocol Support

The BEQ protocol support is both a communications mechanism and a process-spawning mechanism. To use the BEQ protocol support, the client and server must be on the same system. If a network service name is not specified, either directly by the user on the command line or Login screen or indirectly by an environment variable such as ORA_DFLT_HOSTSTR, then the BEQ protocol support is used. In this case, the BEQ protocol support always uses a dedicated server and the shared server model is never used. This dedicated server is started automatically by the BEQ protocol, which waits for the server process to start and attach to an existing System Global Area (SGA). If the startup of the server process is successful, the BEQ protocol support then provides inter-process communication through VMS mailboxes.

An important feature of the BEQ protocol support is that is does not require a listener for its operation. The protocol support is linked into the client tools and directly starts its own server process without outside interaction. However, you can only use the BEQ protocol support when the client program and Oracle9*i* are installed on the same system. The BEQ protocol support is always installed and always linked to all client tools and to the Oracle9*i* server.

10.4 IPC Protocol Support

The IPC protocol support is similar to the BEQ protocol support in that it can only be used when the client program and the Oracle9*i* server are installed on the same system. The IPC protocol support differs from the BEQ protocol support in that it can be used with Oracle Shared Server configurations. The IPC protocol support requires a listener for its operation. The IPC protocol support is always installed and always linked to all client tools and to Oracle9*i*.

10.4.1 Specifying a IPC ADDRESS

The IPC protocol support connection parameters are part of the ADDRESS keyword-value pair. The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file. You can enter the following parameters in any order:

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

Table 10–3 describes the syntax for IPC protocol connections parameters.

Table 10–3 Syntax for IPC Protocol Connection Parameters

Parameter	Description
PROTOCOL	IPC protocol support to be used. The value is IPC. It is not case sensitive.
KEY	Service name of database or system identifier (SID).

Example 10–1 shows a sample IPC ADDRESS.

Example 10–1 IPC ADDRESS Specifying a Client

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

10.5 TCP/IP Protocol Support

The default `tnslnr` port in OpenVMS is 1521.

10.5.1 Specifying a TCP/IP ADDRESS

The TCP/IP protocol connection parameters are part of the ADDRESS keyword-value pair. The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file. You can enter the parameters in any order:

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

Table 10–4 describes the syntax for the TCP/IP protocol connection parameters.

Table 10–4 Syntax for TCP/IP Protocol Connection Parameters

Parameter	Description
PROTOCOL	The protocol support to be used. The value is TCP. It is not case sensitive.
HOST	The host name or the host IP address.
PORT	The TCP/IP port.

Example 10–2 shows a sample TCP/IP ADDRESS.

Example 10–2 TCP/IP ADDRESS Specifying a Client

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

You can specify the last field by name, for example, (PORT=listener).

10.6 Oracle Enterprise Manager

Use the `oratclsh` executable to debug your Tcl scripts. Before executing `oratclsh`, set the TCL_LIBRARY environment variable to specify the `ORA_ROOT:[network.agent.library]` directory.

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

Part III

Appendixes

Part 3 of this guide consists of the following appendixes:

- Appendix B, "Logical Names and Parameters"
- Appendix A, "Messages and Codes"
- Appendix C, "HP OpenVMS Alpha Process Control"
- Appendix D, "Running Oracle Text and Oracle Spatial Demonstrations"

B

Logical Names and Parameters

This appendix provides information about Oracle9*i* logical names and utilities, and the default and recommended values for various initialization parameters. Refer to the *Oracle9i Database Administrator's Guide* for general information about all the initialization parameters.

This appendix contains the following topics:

B.1 Oracle9*i* Logical Names

During installation, several logical names are set up. These assignments will be referenced via the `ORA_ROOT:[bin]ora_db_logicals.com` file that is referenced whenever you start up, upgrade, link, or relink Oracle9*i* or other Oracle products.

B.1.1 Process Quota Logical Names

If you don't set quotas for Oracle's background processes, Oracle9*i* uses its own formulas to determine how to set the quota logical names. Table B–1 shows each quota logical name, along with the minimum and maximum values that you can use if you are setting the logical names yourself, as well as the current formula.

Table B–1 Oracle Quota Limits

Quota Logical Name	Minimum	Maximum	Default
PQL$_ASTLM	0	65536	500
PQL$_BIOLM	0	4096	0
PQL$_BYTLM	1024	1024*1024*1024	4000000
PQL$_CPULM	0	0	0
PQL$_DIOLM	0	4096	0
PQL$_ENQLM	256	16000000	50000
PQL$_FILLM	100	65535	30000
PQL$_JTQUOTA	0	0	0
PQL$_PGFLQUOTA	20480	4,194,304	200000
PQL$_PRCLM	0	20	10
PQL$_TQELM	10	2048	100
PQL$_WSDEFAULT	2048	100000	8000
PQL$_WSEXTENT	2048	1,048,576	100000 (Refer to Table B–2 below)
PQL$_WSQUOTA	2048	1,048,576	20000 (Refer to Table B–2 below)
PQL$_LISTEND	0	0	0

In Table B–2, note the following components of the logicals described in Table B–1, and their values or formulas used to calculate the values:

Table B–2 Components of Oracle Logical Names

Calculation Component	Value or Formula
COMFORT_ZONE	2.5 MB
P0_DYNAMIC_SIZE	Process private storage + room for expansion. 20 MB
P0_IMAGE_SIZE	30 MB size of P0 image.
P0_TABLE_SIZE	`Size of page tables needed to map SGA.`
PQL$_PGFLQUOTA	PAGE_TABLE_SIZE(SGA)+P0_DYNAMIC_SIZE+ COMFORT_ZONE.
PQL$_WSEXTENT	If backing file used: PAGE_TABLE_SIZE (SGA) + 4 * P0_IMAGE_SIZE + P0_DYNAMIC_SIZE + COMFORT_ZONE))/512 + 1 Without backing file used: 4 * (P0_IMAGE_SIZE + P0_DYNAMIC_SIZE + - COMFORT_ZONE))/512 + 1
PQL$_WSQUOTA	If backing file used: (SGA_SIZE/512 + PAGE_TABLE_SIZE (SGA_SIZE) + 4 * (P0_IMAGE_SIZE (PAGE_IMAGE_SIZE) + .6 * (P0_DYNAMIC_SIZE) + COMFORT_ZONE))/512 +1 Without backing file used: PAGE_TABLE_SIZE (SGA_SIZE) + 4 * (PAGE_TABLE_SIZE (P0_IMAGE_SIZE) + .6 * (P0_DYNAMIC_SIZE) + COMFORT_ZONE))/512 +1

For more information about modifying Oracle process quotas through logical names, refer to "Process Quotas" on page 9-15 in this guide.

B.2 System-Dependent Initialization Parameters

All parameters used in init.ora require an equal sign (=). For example, DB_BLOCK_SIZE = 8192 is correct.

B.2.1 BACKGROUND_ DUMP_DEST

Purpose:

Identifies the directory where the trace files created by the detached Oracle9*i* processes are sent.

Recommended Value:

None.

Default Value:

`ORA_ROOT:[admin.<dbname>.bdump]`

Distributed Value:

None.

B.2.2 DB_BLOCK_SIZE

Purpose:

Identifies size (in bytes) of Oracle9*i* database blocks and the database buffers in the SGA.

Recommended Value:

A binary multiple of 512 bytes (which is the HP OpenVMS Alpha I/O block size). The maximum value is 32768.

Default Value:

8192

Distributed Value:

None

B.2.3 LOG_ARCHIVE_DEST

Purpose:

Specifies a default text string to indicate the location and name of the disk file when archiving log files. Archiving directly to tape is not supported.

Recommended Value:

Any valid disk filename.

Default Value:

ORA_ARCHIVE

Distributed Value:

None

B.2.4 LOG_ARCHIVE_ FORMAT

Purpose:

Specifies a default filename format for archived redo log files. LOG_ARCHIVE_FORMAT is appended to the string specified in the LOG_ARCHIVE_DEST parameter.

The redo log file format specifications can contain variables that are substituted with a unique archived redo log file name. Refer to the "Recovering a Database" chapter of the *Oracle9i Database Administrator's Guide* for more information on these variables

Recommended Value:

Any valid file name format

Default Value:

ARCH%T_%S.ARC

Distributed Value:

None

B.2.5 PRE_PAGE_SGA

Purpose:

Determines whether the SGA pages will be paged into each user's working set at connect time. This parameter can be manipulated to reduce page faults.

Recommended Value:

Define this parameter as TRUE if the current system load has not produced a high rate of page faults.

Default Value:

False

Distributed Value:

None

B.2.6 SHARED_POOL_SIZE

Purpose:

Determines the size of the shared pool. The shared pool contains shared cursors and stored procedures.

Recommended Value:

Larger values of this parameter improve performance in multi-user systems. Smaller values use less memory. This parameter's minimum is 300 KB and its maximum is determined by the size of your SGA. Although there are no SGA size limitations on HP OpenVMS Alpha, the minimum value is 15 MB.

Default Value:

112 MB

Distributed Value:

None

B.2.7 SORT_AREA_SIZE

Purpose:

Identifies the size of real memory (in bytes) that will be available for sorting processes.

Recommended Value:

The amount of real memory that you can reasonably expect to have available for sorting. For example, on a system with 256 MB of real memory, with 1/8 available to sort processes and 4 sorts occurring at the same time, you might set this parameter to 256/8/4 = 8 MB.

Default Value:

Generally, a larger size only improves the efficiency of large sorts. However, the default is usually fine for most database operations.

Distributed Value:

None

B.2.8 USER_DUMP_DEST

Purpose:

Identifies the location to which trace files created by user processes are sent.

Recommended Value:

None.

Default Value:

`ORA_ROOT:[admin.<dbname>.udump]`

Distributed Value:

None

B.2.9 VLM_BACKING_STORAGE_FILE

Purpose:

Determines whether to use a backing file for the SGA instead of using a memory resident global section.

Recommended Value:

FALSE, unless there is a well understood need to allow the SGA to page.

Setting this parameter to TRUE will disable the use of OpenVMS Fast I/O, slow process startup, and in most cases reduce performance.

Default Value:

FALSE

Distributed Value:

None

C

HP OpenVMS Alpha Process Control

This appendix presents some useful tips about managing your HP OpenVMS Alpha processes. For more information about a specific application development tool, refer either to the product's generic documentation or to the product's chapter in this guide.

Your Hewlett-Packard documentation contains additional information on these topics.

This appendix contains the following topics:

C.1 Interrupting and Terminating Oracle Operations

The following sections explain how to cancel an operation without aborting, how to cancel an operation with the option to continue and how to disable the control keys.

C.1.1 Cancelling without Aborting the Oracle Image

To cancel an operation without aborting the Oracle image, press [CONTROL]-C. The current query is cancelled. After pressing [CONTROL]-C, you might need to press the [RETURN] key to bring the prompt back (particularly when you are using command line tool such as SQL*Plus).

C.1.2 Cancelling with the Option to Continue

You can also terminate any Oracle operation by pressing [CONTROL]-Y. This returns the DCL prompt ($) with a message that Oracle has been interrupted.

To continue your Oracle session, type "CONTINUE." To terminate the session, type "EXIT."

Typing "EXIT" or any other OpenVMS command cancels the query and runs down the tool. Typing "EXIT" causes a noticeable delay before the DCL prompt returns or before the requested HP OpenVMS Alpha command executes because a partial shutdown of your Oracle session occurs.

C.1.2.1 Known Limitations in Client-Server Connections

Any client tool you are using is connected to Oracle through a client-server Oracle Net connection (for example, by using the VMS Mailbox). If the query is cancelled, the tool shuts down, but no message is sent from the tool to the server to tell it to terminate the server session. On sensing that a client tool has aborted, many Oracle Net protocols send a message to the server that causes the server session to terminate, but some do not. In this case, the server sessions continue indefinitely until Oracle kills the session because the user session has a finite idle-time limit, the system shuts down, or some watchdog process on the node kills the idle process. This is a known limitation.

If a user's terminal unexpectedly disconnects from OpenVMS while engaged in a client-server connection to Oracle (connecting to HP OpenVMS Alpha through a LAT terminal or a personal computer is turned off), the client tool is aborted through the [CONTROL]-Y and "EXIT" method. The current query, if any, is cancelled and the tool does a partial shutdown.

C.1.3 Disabling Control Keys

To disable the control keys, enter the command:

```
$ SET TERM/PASTHRU
```

C.2 Running Oracle Programs as Detached Processes

Sometimes you might want to run programs as detached processes; for example, you might want to run a Pro*C program while you are logged into SQL*Plus or while doing work unrelated to Oracle.

A detached process does not inherit the logical names that its parent has. Consequently, when a program executable is passed to the detached process, the detached process will abort because it cannot find the logical names referenced by the program.

You can work around this problem by invoking the login process LOGINOUT, which maps DCL into the detached process's virtual space. This can execute a command procedure to run the program in the detached process. The command file should:

1. Set up the proper execution environment by defining the referenced logical names, symbols, and defaults.
2. Invoke the program to be executed. For example:

   ```
   $ RUN/DETACH/INPUT=TEST.COM/OUTPUT=TEST.LOG SYS$SYSTEM:LOGINOUT
   ```

   where TEST.COM is:

   ```
   $ @DISK$TEST:[ORACLE9i]ORAUSER SID
   $ RUN <myprog>.EXE
   ```

You might need to include certain process quotas to map DCL into the detached process's virtual space. Refer to Hewlett-Packard's documentation for more information.

D

Running Oracle Text and Oracle Spatial Demonstrations

This chapter contains information on running the Oracle Text and Oracle9*i* Spatial demonstrations.

It contains the following topics:

D.1 Oracle Text

See the `ORA_ROOT:[ctx.sample.api]index.html` file for information on the Oracle Text code samples.

> **See Also:** the *Oracle Text Reference,* for more information on Oracle Text.

D.2 Oracle Spatial

Refer to the `ORA_ROOT:[md.doc]readme.txt` file for information on the Oracle9*i* Spatial demonstration. Refer to the *Oracle Spatial User's Guide and Reference* for information on Oracle9*i* Spatial.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W