

Oracle8 for Alpha OpenVMS

Server and Tools Administrator's Guide

Release 8.0.5

October, 1999

Part No. A77041-01

ORACLE®

Oracle8 for Alpha OpenVMS Server and Tools Administrator's Guide, Release 8.0.5

Part No. A77041-01

October 1999

Copyright © 1996, 1999, Oracle Corporation. All rights reserved.

Primary Author: Anjana Suparna Sriram

Contributors: Sudhakar Chitageri, Charles Congdon, DJ Cover, Kurt Engeleiter, Phil Goerl, David Hayter, Gary Huffman, Pierre Krabbendam, Thomas Leah-Martin, Tony Lekas, Saar Maoz, Thomas Arnold, Alice Mulder, Tony Purmal, David Schwab, Peter Shih, Bob Smith, Regina Rohr, Gary Hodson, John Sobecki

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent and other intellectual and industrial property laws. Reverse engineering, disassembly or decompilation of the Programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government Agency or anyone licensing or using the Programs on behalf of the US Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are 'commercial computer software' and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are 'restricted computer software' and use, duplication, and disclosure of the Programs should be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the Program.

Contents

Send Us Your Comments	ix
------------------------------------	-----------

About this Guide	xi
-------------------------------	-----------

Part I Administering the Oracle8 Enterprise Edition

1 Introduction to Oracle8 on Alpha OpenVMS

New Features on Alpha OpenVMS.....	1-2
DBJAVA (JDBC, Java Database Connectivity)	1-2
Oracle Parallel Server.....	1-2
Oracle8 on Alpha OpenVMS.....	1-3
Oracle8 Code	1-3
Oracle8 Instances	1-4
Logical NamesDatafiles: Locations and Identifying by	1-8

2 Setting Up Oracle8 Users

Granting Access to Oracle8 Users.....	2-2
Using the Oracle8 Password Utility	2-3
Ending a User's Session.....	2-5

3 Starting Up and Shutting Down Oracle8

Starting Up the Oracle8 Enterprise Edition.....	3-2
Before Start Up.....	3-2
Starting Oracle8 via ORACLEINS.....	3-3

Starting Oracle8 via STARTUP Files.....	3-4
Starting Oracle8 via Server Manager	3-4
Starting Oracle8 Remotely via Server Manager from an OpenVMS Client.....	3-6
Starting Oracle8 Remotely via Server Manager from a Windows PC Client	3-8
Starting Oracle8 Automatically	3-10
Shutting Down the Oracle8 Enterprise Edition	3-12
Stopping Oracle Users Before Database Shutdown.....	3-12
Shutting Down Oracle8 using ORACLEINS	3-14
Shutting Down Oracle8 using the SHUTDOWN File	3-15
Shutting Down Oracle8 using Server Manager.....	3-16
De-installing Shareable Images	3-16

4 Managing Instances

Managing Single Instances	4-2
Components of Instances.....	4-2
Controlling Instances	4-2
Deleting Instances.....	4-2
Adding and Enabling Instance Threads.....	4-3
Preparing an Instance for Parallel Query	4-4

5 Managing the Database

Server Manager and SQL*Net.....	5-2
Starting Server Manager	5-2
SET ECHO and SET TERMOUT Functionality in SVRMGR	5-2
Creating Database Links	5-3
Creating Multiple Control Files.....	5-3
Managing Database Files	5-4
Using Commands to Manage Database Files	5-4
Adding Files	5-5
Renaming Files.....	5-5
Moving Tablespace Files.....	5-6
Moving Redo Log Files	5-7
Database Verification Utility and Other Useful Utilities	5-8
Database Verification Utility.....	5-8
Other Useful Utilities	5-9

Debugging Database Processes with ORAMBX	5-9
6 Backing Up and Archiving Your Database	
Archiving Redo Log Files.....	6-2
Specifying Archive Destinations	6-2
Archiving Automatically.....	6-3
Archiving Manually	6-4
Backing Up the Database	6-5
Backing Up a Closed Database	6-5
Backing Up an Open Database	6-6
Backing Up Data Structures and Definitions	6-8
Exporting to and Importing from Multiple Tapes.....	6-9
Exporting to Multiple Tapes	6-10
Importing from Multiple Tapes.....	6-10
7 Recovering Your Data	
Overview	7-2
Recovering from Instance Failure.....	7-2
Recovering from Media Failure	7-3
Media Recovery	7-3
Restoring from an Export File.....	7-4
8 Optimizing Oracle8	
Tuning Memory Usage for the Oracle8 Enterprise Edition.....	8-2
Adjusting INIT.ORA Parameters	8-2
Modifying Alpha OpenVMS Process Quotas.....	8-3
Adjusting the SGA.....	8-4
Installing Products in Shared Memory	8-5
Installing Oracle Products	8-5
Running ORA_INSUTL	8-6
Reducing Database Fragmentation	8-7
9 Trace Files	
Using Trace Files	9-2

Specifying Trace File Directories	9-2
Identifying Trace Files	9-3
No Need to Format.....	9-4
INIT.ORA Parameter for Creating World-readable Trace Files	9-4

Part II Using and Administering the Oracle Tools

10 Oracle Programmatic Interfaces

Programmatic Interfaces.....	10-2
SQL*Module	10-2
Directory Structure	10-2
PROGINT Branch of the ORA_ROOT Directory Structure.....	10-3
Precompiler Executable Files	10-3
Precompiler Include Files	10-3
Precompiler Oracle Call Interface (OCI) Files	10-4
Precompiler Demo Files.....	10-4
Precompiling.....	10-5
Syntax	10-5
Example.....	10-6
Guidelines and Restrictions	10-7
Using the Alpha OpenVMS Debugger	10-7
Using Event Flags.....	10-7
Migrating Applications Developed with Pro*C Compilers	10-7
Compiling.....	10-7
Compiler Options Used to Compile Oracle8.....	10-7
Floating Point Formats.....	10-8
..... Pro*COBOL	10-8
Linking.....	10-8
Syntax	10-9
Example.....	10-10
Including Option Files	10-10
Guidelines	10-10
Using the Oracle Call Interface (OCI) Routines	10-12
Guidelines	10-12
CDA/LDA Structure Information	10-13

Linking OCI Programs Written in C.....	10-13
Linking OCI Programs Written in Other Languages	10-14
Unexpected Link Errors.....	10-15
Data Areas and Datatypes.....	10-15
Binary Integers.....	10-15
Using Literals as Call Arguments	10-15
Optional or Missing Parameters.....	10-16
Using Event Flags	10-16

11 SQL*Plus

Warning Message When Invoking SQL*Plus	11-2
Interrupting SQL*Plus.....	11-2
Installing Help Tables.....	11-2
Running System Commands.....	11-3
Passing Parameters to SQL*Plus.....	11-3
Passing Parameters to SQL*Plus from OpenVMS.....	11-3
Passing Parameters to SQL*Plus Interactively from a User	11-4
Passing Parameters to SQL*Plus from an Input File	11-5
Passing Values from SQL*Plus	11-6
Passing a Numeric Value	11-6
Passing Multiple Values	11-7
Using Profile Files	11-7
Types of Files.....	11-7
Order of Execution	11-8
Creating the Files.....	11-8
Using the OpenVMS Editor from SQL*Plus.....	11-9
Converting SQL*Plus Output Files.....	11-9
Interpreting Output from the SQL*Plus TIMING Command	11-10
Exit Status within DCL.....	11-10

Part III Appendices

A Logical Names and Parameters

Oracle8 Logical Names	A-2
-----------------------------	-----

Where Logical Names Are Defined	A-2
Alphabetical Listing	A-3
Process Quota Logical Names.....	A-6
System-Dependent Initialization Parameters	A-8
BACKGROUND_ DUMP_DEST	A-9
CONTROL_FILES.....	A-9
DB_BLOCK_SIZE	A-9
LOG_ARCHIVE_DEST	A-10
LOG_ARCHIVE_ FORMAT.....	A-10
PRE_PAGE_SGA	A-10
SHARED_POOL_SIZE.....	A-11
SORT_AREA_SIZE.....	A-11
USER_DUMP_DEST	A-11
VLM_BACKING_STORAGE_FILE.....	A-12

B Messages and Codes

C Alpha OpenVMS Process Control

Interrupting and Terminating Oracle Operations	C-2
Cancelling without Aborting the Oracle Image	C-2
Cancelling with the Option to Continue	C-2
Disabling Control Keys.....	C-3
Running Oracle Programs as Detached Processes.....	C-3

D Alpha OpenVMS Supplement to the Generic Documentation Set

Server Guides	D-2
Utilities Guide	D-14

Index

Send Us Your Comments

Oracle8 for Alpha OpenVMS Server and Tools Administrators's Guide, Release 8.0.5

Part No. A77041-01

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 chapter, section, and page number (if available). You can send comments to us in the following ways:

- E-mail - your_product@us.oracle.com
- FAX - (650) 506-7000. Attn: Documentation Manager
- Postal service:
Compaq Products Documentation Manager
Oracle Corporation
Mailstop 659105
Redwood City, CA 94065

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

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

About this Guide

This guide provides instructions on how to install and configure Oracle8 Enterprise Edition and related Oracle products on Alpha OpenVMS systems. It guide supplements information found in the Oracle8 and Oracle8 Enterprise Edition Documentation Set.

Oracle8 and Oracle8 Enterprise Edition

Unless noted otherwise, features and functionality described in this document are common to both Oracle8 and Oracle8 Enterprise Edition.

Audience

This guide is written for system administrators or database administrators (DBAs), who administer Oracle8 products on Alpha OpenVMS systems. This guide assumes that you have a fundamental knowledge of the Alpha OpenVMS operating system; it does not document any features of Alpha OpenVMS except those that affect or are affected by Oracle8.

The README files in the Oracle product directories contain the latest information about the Alpha OpenVMS production releases of Oracle8 and related products for Alpha OpenVMS.

The purpose of this guide is to help you administer the Oracle8 Enterprise Edition and its application development tools.

For the Latest Information

For the latest information about Alpha OpenVMS production releases of Oracle8 and related products, see the README files in the Oracle product directories.

Install Only Licensed Products

You are entitled to install and use only those products for which you have a current Oracle license agreement.

How this Guide Is Organized

This guide is organized as follows:

PART I: Administering the Oracle8 Enterprise Edition

This part covers activities that are related to maintaining the database once the Oracle8 Enterprise Edition has been installed and the instances accessing the database have been set up.

Chapter 1: Introduction to Oracle8 on Alpha OpenVMS

This chapter introduces Oracle products and gives an overview of Oracle8 system on Alpha OpenVMS.

Chapter 2: Setting Up Oracle8 Users

This chapter describes how to grant access to Oracle8 and how to set up specific tools.

Chapter 3: Starting Up and Shutting Down Oracle8

This chapter describes how to start up and shut down Oracle8.

Chapter 4: Managing Instances

This chapter describes how to define, delete, and control single and parallel instances.

Chapter 5: Managing the Database

This chapter describes how to manage the files that make up the physical database and other topics related to managing the database.

Chapter 6: Backing Up and Archiving Your Database

This chapter describes how to back up and archive your database.

Chapter 7: Recovering Your Data

This chapter describes how to recover previously backed up data.

Chapter 8: Optimizing Oracle8

This chapter describes how to tune Oracle8 and how to identify, monitor, and correct situations that cause problems while administering the database.

Chapter 9: Trace Files

This chapter describes how to set up and use trace files.

PART II : Using and Administering the Oracle Tools

This part presents the tools in separate chapters, organized alphabetically.

Chapter 10: Oracle Programmatic Interfaces

This chapter explains how to use the Oracle Programmatic Interfaces and SQL*Module in the Alpha OpenVMS environment.

Chapter 11: SQL*Plus

This chapter explains how to use SQL*Plus in the Alpha OpenVMS environment.

PART III : Appendices

This part includes supplementary information.

Appendix A: Logical Names and Parameters

This appendix provides information about Oracle8 logical names and Alpha OpenVMS-specific INIT.ORA parameters that are defined differently for different operating systems.

Appendix B: Messages and Codes

This appendix explains the means of error messages you might see while using Oracle products on Alpha OpenVMS.

Appendix C: Alpha OpenVMS Process Control

This appendix explains how to control Oracle Server processes on Alpha OpenVMS.

Appendix D: Alpha OpenVMS Supplement to the Generic Documentation Set

This appendix gives Alpha OpenVMS specific information for each cross-reference from the generic documentation set.

Conventions Used in this Guide

This section explains the following:

- Syntax

Syntax

This guide uses the following conventions:

monospaced font	Monospaced font is used to represent information displayed on a terminal or monitor or entered on a keyboard. For example, menu screens that are displayed during the Oracle8 installation procedure are represented in this guide with monospaced font.
UPPERCASE	Uppercase in monospaced font represents a command name. Enter the text exactly as shown.
UPPERCASE	Uppercase words within the text refer to command names.
<variable>	<lowercase italicized words> in monospaced font enclosed by angle brackets represent a variable on the command line. Substitute an appropriate value.
<variable>	<lowercase italicized words> enclosed by angle brackets within the text refer to variable names.

Related Products and Documents

This section explains related products and related documents.

Related Products

The following is a list of Oracle products, utilities, and options that are available with Oracle8:

- Advanced replication option
- ConText option
- Data partitioning option
- Distributed database option
- DBJAVA (JDBC, Java Database Connectivity)
- Image Cartridge

- National Language Support or Multilingual Option
- Object Option
- Oracle Parallel Server
- Oracle Intelligent Agent
- Oracle Terminal
- Parallel query option
- Programmatic Interfaces (Oracle precompilers)
- Server Manager
- Spatial Data Cartridge
- SQL*Net 8.0.5
- SQL*Plus
- Time Series Cartridge
- UTIL
- Visual Information Retrieval Data Cartridge

Note: The Advanced Replication, Object Support, Spatial Data, and Data Partitioning options are available only in the Oracle8 Enterprise Edition release. They are not included in the Oracle8 release.

Related Documents

This documentation set includes the following guides:

- Oracle8 for Alpha OpenVMS Server and Tools Administrator's Guide is this guide.
- The Oracle8 for Alpha OpenVMS Installation Guide describes installing, upgrading, or migrating Oracle8.
- SQL*Net for Alpha OpenVMS Configuration and User's Guide describes how to configure, maintain, and use SQL*Net on Alpha OpenVMS.
- Release Notes (varies by release)
- README files

For more information on the Oracle8 Enterprise Edition, refer to the following generic documentation:

- The Oracle8 Server Administrator's Guide, Volume I, II, and III contain detailed information about administering Oracle8.
- The Oracle8 Application Developer's Guide contains specific information required to develop applications for Oracle8.
- Oracle8 Server Concepts contains generic information about the Oracle8 Server and describes features and maintenance options available.
- Oracle8 Server Distributed Systems, Volume I: Distributed Data describes how to use the distributed data option.
- Oracle8 Server Distributed Systems, Volume II: Replicated Data describes how to use the advanced replication option.
- Oracle8 Server Messages lists all of the messages and codes that Oracle can return.
- The Oracle8 Server Reference describes the Oracle data dictionary tables, initialization parameters, national language support features, and so on.
- The Oracle8 Server SQL Reference contains generic information about the Oracle8 SQL language.
- Oracle8 Server Tuning shows you how to diagnose performance problems and take corrective action.
- The Oracle8 Server Utilities describes the auxiliary utilities provided with the Oracle Server, such as SQL*Loader, the Import utility, and the Export utility.

- Oracle8 Parallel Server Concepts and Administration describes the special features of Oracle8 running on a loosely-coupled system.
- The Oracle Server Manager User's Guide describes how to use Server Manager, a graphical user interface for doing administrative tasks.

Migrating from Oracle Version 7

For more information about migrating from Oracle Version 7 to Oracle8, see the Oracle8 Server Migration Guide.

Customer Support Information

(Please copy this page and distribute within your organization as necessary.)

For Oracle Worldwide Oracle Support Services (OSS), contact your local number. (The hours are detailed in your support contract.) _____

Please prepare the following information before you call, using this page as a checklist:

- ☐ Your Customer Support Identification (CSI) number if applicable, or full contact details, including any special project information
- ☐ The complete release numbers of the Oracle8 Enterprise Edition and associated products (for example, Oracle8 Enterprise Edition release 8.0.5 or Oracle Forms release 4.5.6.3.2).
- ☐ The hardware type on which the problem occurs (for example, Compaq Alpha)

- ☐ The operating system name and release number (for example, Alpha OpenVMS 7.1)_____
- ☐ Details of error codes and associated descriptions. Please write these down as they occur, since they are critical in helping OSS to quickly resolve your problem. _____
- ☐ A full description of the issue, including:
- ☐ **What** - What happened? For example, the command used and result obtained.
- ☐ **When** - When did it happen? For example, time of day, or after a certain command, or after an O/S upgrade.
- ☐ **Where** - Where did it happen? For example, on a particular system or within a certain procedure or table.

- ☐ **Extent** - What is the extent of the problem? For example, production system unavailable, or moderate impact but increasing with time, or minimal impact and stable.

Note: Keep in mind what *did not* happen, as well as what *did* happen. This type of information can help OSS to more quickly resolve your problem.

- ☐ Keep copies of any trace files, core dumps, and redo log files recorded at or near the time of the incident, since OSS will need these to further investigate your problem.

For installation-related problems please have the following information available:

- ☐ Error returned by the installation procedure and/or Alpha OpenVMS _____

Your Comments Are Welcome

We value and appreciate your comments as an Oracle user and reader of the manuals. As we write, revise, and evaluate our documentation, your opinions are the most important input we receive. At the back of our printed manuals is a Reader's Comment Form, which we encourage you to use to tell us what you like and dislike about this manual or other Oracle manuals. If the form is not available, please use the following address, phone number, or FAX number.

Compaq Products Division
Publications Manager
Oracle Corporation
500 Oracle Parkway
Box 659105
Redwood Shores, CA 94065
Phone: (650) 506-7000
FAX: (650) 506-7361

Part I

Administering the Oracle8 Enterprise Edition

Introduction to Oracle8 on Alpha OpenVMS

This chapter introduces the Oracle8 products in the Alpha OpenVMS environment. It consists of the following sections:

- Overview of Oracle products on Alpha OpenVMS
- Oracle8 on Alpha OpenVMS

New Features on Alpha OpenVMS

Oracle8 Alpha OpenVMS release 8.0.5 provides the following new features:

- DBJAVA (JDBC, Java Database Connectivity)
- Oracle Parallel Server

DBJAVA (JDBC, Java Database Connectivity)

DBJAVA allows Java programs to access the Oracle8 database.

Oracle Parallel Server

Oracle Parallel Server mode is now supported with the Oracle8 and Oracle8 Enterprise Edition Release 8.0.5.1 for Alpha OpenVMS 7.2 or higher.

Parallel Server mode can be used on any cluster node, and can take advantage of any cluster communications media, including Memory Channel, FDDI, and Galaxy Shared Memory.

Oracle Parallel Server mode uses the Oracle Group Membership Services facility. Each node which runs an OPS database must also run the OGMS Daemon. Please see the Oracle8 Parallel Server Concepts and Administration guide, version 8.0.4 or later, for more information on Oracle GMS.

Please refer to the READMEVMSOPS.DOC in the ORA_RDBMS directory for more information about Oracle Parallel Server.

Oracle8 on Alpha OpenVMS

Figure 1-1 shows an overview of the Oracle8 Enterprise Edition architecture.

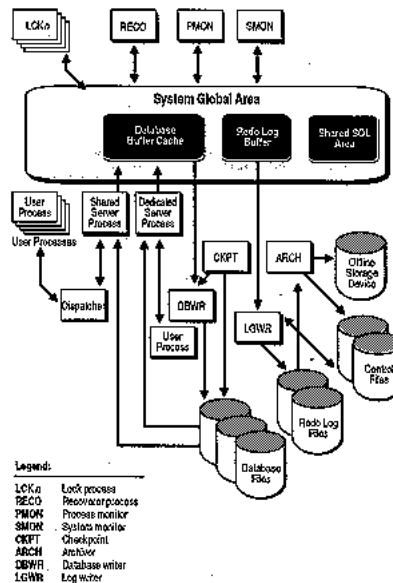


Figure 1-1 Oracle8 Enterprise Edition architecture

Oracle8 Code

Oracle8 code consists of several object libraries that are used to form the Oracle8 image during installation; you must link this image with the DEC runtime libraries to produce the executable Oracle8 code.

The code also consists of shared Oracle client image linked during installation. For more information about shared Oracle client image, refer to the *Oracle8 for Alpha OpenVMS Installation Guide*.

Code for the Oracle8 Enterprise Edition is built to use 64-bit pointers to support very large SGAs. 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.

ORACLE.EXE File

When the Oracle8 Enterprise Edition is linked and installed, these routines reside as shareable code in Alpha OpenVMS global memory.

Occasionally, you will need to relink the image, such as when new code is distributed or when a new release of Alpha OpenVMS is installed.

Oracle8 Instances

An Oracle8 instance is a combination of Oracle Server processes and memory buffers, as shown in Figure 1-1 .

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 ORA_SID. The SID must be assigned to the logical name ORA_SID before the instance starts.

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 Server Manager 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.

Oracle8 release 8.0.5 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.

By default the SGA is not pageable. Once the SGA is created, the system cannot reclaim any of the memory that the SGA uses.

Additional Information: Refer to the *Oracle8 for Alpha OpenVMS Installation Guide* for your platform for information about making the SGA pageable.

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.

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 3.6 MB.

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.

The number of blocks that can be maintained in the buffer pool is determined by the initialization parameter `DB_BLOCK_BUFFERS`. For more information, see the *Oracle8 Server Administrator's Guide*.

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. For more information, see the *Oracle8 Server Administrator's Guide*.

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:[DB_<dbname>] directory, where <dbname> is the name you assign to the database. You can specify any directory for the first data file, and this directory does not necessarily need to be under ORA_ROOT. 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 Oracle8 Enterprise Edition block size. For example, if the blocks are 2 KB, then the cluster size should be 4 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 4 KB cluster is an 8-block cluster.

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

Storing Changes to the Database

Changes made to the database are logged in the shared database buffers 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 buffers 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 OpenVMS blocks.

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 cannot perform media recovery. You must use ARCHIVELOG mode in order to recover from media failure.

When a redo log file fills, the DBA must back up 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 back up 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.

The redo log files must be at least 50Kb (100 OpenVMS blocks). 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 2000 Kb each by default; you can alter this size and specify different file names during the installation procedure if you want.

For more information, refer to the *Oracle8 Server Administrator's Guide* and to Chapter 6 and Chapter 7 in this guide.

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 or group level logical names (based on whether you used system or group installation) to name the devices where the database and redo log files reside, and that you specify full directory and filename paths for these files. You may fully specify the control file names with logical names, as with the ORA_CONTROL1 and ORA_CONTROL2 logical names.

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 *Oracle8 Server Administrator's Guide*.

Logical NamesDatafiles: 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 Oracle8 account must have write access to the location of the datafiles.
- Datafiles cannot be put in the root level directory of a disk.

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.

Setting Up Oracle8 Users

This chapter assumes that you have installed Oracle8 and have created a database and started an instance. This chapter describes how to set up Oracle8 users, as well as how to set up certain application development tools. Only those tools having special requirements are described.

The following topics are presented:

- Granting Access to Oracle8 Users
- Using the Oracle8 Password Utility
- Ending a User's Session

Granting Access to Oracle8 Users

When the Oracle8 Enterprise Edition is installed with the Group option, all Oracle8 users must belong to the same group that includes the Oracle8 account. Otherwise, the Oracle8 Enterprise Edition can be accessed by users in any UIC group.

To grant users access to Oracle8, 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_<dbname> <sid> -  
    <setup_node>
```

device	Disk device or logical name where the ORACLE account resides
directory	Directory path to the database-specific directory where the appropriate ORAUSER_<dbname>.COM file resides
dbname	Name of the database
sid	SID of the Oracle8 instance that this user should access. This is a qualifier that is optional depending on the circumstances of the instance setup.
setup_node	Node where the instance was set up. This qualifier is optional depending on the circumstances of the instance setup.

For example:

```
$ @DISK$MIS:[ORACLE.V8.DB_MIS]ORAUSER_MIS MKT1 HARPO
```

Instead of completing step 1, you can define a symbol in the system-wide login procedure (typically, SYLOGIN.COM) that executes a particular ORAUSER_<dbname>.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:

```
$ HARPO == -  
"@DISK$MIS:[ORACLE.V8.DB_MIS]ORAUSER_MIS MKT1 HARPO"
```

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, see the “Setting Up the Oracle Accounts” chapter of the *Oracle8 for Alpha OpenVMS Installation Guide*.

3. Create the Oracle8 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 *Oracle8 Server Administrator's Guide*.
4. If you have a user who uses the SVRMGRL utility to start up or shut down an Oracle8 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, see Chapter 2 in the *Oracle8 for Alpha OpenVMS Installation Guide*.

Using the Oracle8 Password Utility

On Alpha OpenVMS, you do not create a password file as part of the Oracle8 installation procedure. Instead, you must use the Oracle password utility.

With the password utility, you still need an OpenVMS account for each OSOPER and OSDBA Oracle8 account, but with version 7.1 and later, OpenVMS passwords are no longer used for connections.

To use the password utility, use the following steps:

1. Set default to ORA_DB
2. Invoke ORAPWD using the following syntax:

```
$ ORAPWD FILE=<file> PASSWORD=<password> ENTRIES=<users>  
where
```

file	You must enter the name of the password file.
password	You must enter a password for SYS and INTERNAL. The Oracle8 Enterprise Edition uses this password to authenticate a statement such as "CONNECT INTERNAL/<password> AS SYSDBA" or "CONNECT SYS/<password> AS SYSDBA". To log in as SYS into the user schema, use "CONNECT SYS/CHANGE_ON_INSTALL" (or your present setting for the SYS password), since the password in the file can be different from the password already assigned to the SYS account. However, once you bring up the database with the password file in private mode (exclusive), then changing the password of SYS will make the one in the file the same as the one in the database. If you want to authorize local connects or remote connects with secure protocols you don't need to use the password file (if you have the appropriate OS role). Since TCP/IP is not truly a secure protocol, Oracle Corporation strongly recommends that you make use of the password utility for these protocols.
users	The maximum number of OSDBA or OSOPER users that will be allowed to use the database. Since the password file can NOT be expanded, make sure that you take into account any accounts that you will have to create-now or in the future.

3. Define an executive-mode logical name that identifies the password file using the following syntax:

```
$ DEFINE/SYSTEM/EXEC ORA_<sid>_PWFILE <location>:<name>
```

Note: In system startup and shutdown, this logical name needs to be redefined prior to starting Oracle.

For example:

```
$ DEFINE/SYSTEM/EXEC ORA_PAYROL_PWFILE -  
DISK$MIS3:[ORACLE.DB_FIN]FIN_PWD_FILE.PWD
```

Note: Define the logical name in the startup script for each instance of the database, and not in
ORA_DB:ORA_DB_<dbname>.COM or
ORA_DB:ORAUSER_<dbname>.COM.

4. Cycle your database, restarting it in exclusive mode.

Refer to the *Oracle8 Server Administrator's Guide* for information on how to define the passwords for the OSDBA and OSOPER accounts.

5. Once all passwords have been assigned, restart your database in the appropriate mode (EXCLUSIVE).

All non-secure local or remote connections will now use the passwords for the OSDBA and OSOPER accounts as defined in the ORACLE password file. For more information, refer to the *Oracle8 Server Administrator's Guide*.

Ending a User's Session

You can end a user's Oracle8 session in one of two ways:

- Issue the `STOP/ID=<process_id>` command (the recommended method).
- Issue the `ALTER SYSTEM KILL SESSION` command in SQL*Plus or Server Manager.

Note: You should only use **one** of the above methods to end a specific Oracle8 session; you should not issue both commands. If you issue both commands for the same user session and then attempt to `SHUTDOWN IMMEDIATE`, the shutdown will fail.

Starting Up and Shutting Down Oracle8

This chapter describes different ways to start up or shut down the Oracle8 Enterprise Edition. These methods include using ORACLEINS, using STARTUP and SHUTDOWN files, or using Server Manager.

The following topics are presented:

- Starting Up the Oracle8 Enterprise Edition
- Shutting Down the Oracle8 Enterprise Edition

Starting Up the Oracle8 Enterprise Edition

Before you can start the Oracle8 Enterprise Edition, both an instance and a database must exist on your local system. If you did not install the Oracle8 Enterprise Edition, consult the person who did.

This section presents the following topics:

- Before Start Up
- Starting Oracle8 via ORACLEINS
- Starting Oracle8 via STARTUP Files
- Starting Oracle8 via Server Manager
- Starting Oracle8 Remotely via Server Manager from an OpenVMS Client
- Starting Oracle8 Remotely via Server Manager from a Windows PC Client
- Starting Oracle8 Automatically

Before Start Up

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

After rebooting Alpha OpenVMS, you must perform the following steps before starting the Oracle8 Enterprise Edition:

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

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

2. Run the ORA_RDBMS:INSORACLE.COM file.

This file installs the shared global sections that make a shareable ORACLE image known to the system.

The following images are installed:

- ORACLIENT_<image_id>.EXE
- ORACLIENT64_<image.id>.EXE
- ORACLE.EXE

Each of these images must have proper protection when you run INSORACLE. The account where you run INSORACLE.COM must have CMKRNL privilege.

You *must* perform step2 under the following conditions:

- After the ORACLE image has been removed, but before the Oracle8 Enterprise Edition is started
- After running the REMORACLE.COM command file (which de-installs the global sections loaded by INSORACLE.COM)
- Whenever the computer is booted

Note: Running INSORACLE.COM might cause problems with any currently running instance that uses the shareable images that these command files install (for example, the database might go down). Take this into account if you create an instance-specific automatic startup procedure that invokes the INSORACLE file.

Starting Oracle8 via ORACLEINS

To start Oracle8 using ORACLEINS, do the following steps:

1. Run the database-specific ORAUSER file using the following syntax:

```
$ @ORA_DB:ORAUSER_<dbname>.COM <sid> <setup_nodename>
```

2. Run ORACLEINS:

```
$ ORACLEINS
```

3. Select option 3, "Reconfigure existing products, manage the database, or load demo tables," from the Oracle Installation Startup Menu.
4. Press [RETURN] when prompted to specify the root directory.
5. Press [RETURN] when prompted to specify the device where you mounted the distribution medium.
6. Select option 2, "Instance Creation, Startup, and Shutdown Menu" from the Main Menu.

7. Select option 2, "Startup an Existing Instance," from the Instance Creation, Startup, and Shutdown Menu. The following message is displayed:

```
Currently known database SIDs:
[  
list of known SIDs  
]  
Press [RETURN] to quit with no action.  
NOTE: The SID can be a maximum of 6 characters in length.  
What is the SID for the instance to startup?
```

8. Type the SID of the instance that you want to start and press [RETURN]. The instance identified by this SID is started and the database associated with this instance is opened in exclusive mode.

Starting Oracle8 via STARTUP Files

You can also use command files to start Oracle8. The file you execute depends on whether you are running in exclusive or in parallel mode. For more information about instances, see Chapter 4, "Managing Instances" in this guide.

Run the following STARTUP command file for the instance you want to start:

```
$ @ORA_DB:STARTUP_EXCLUSIVE_<dbname>.COM <sid> <setup_nodename>
```

This file is located in the database-specific directory identified by the logical name ORA_DB. When you start up the instance, be sure to specify the SID of the instance and its setup node.

Starting Oracle8 via Server Manager

You can also start an instance of Oracle8 using Server Manager. See the instructions in this manual on setting up Server Manager on your Alpha OpenVMS platform. Refer to the generic (platform-independent) Oracle Server documentation for instructions on using Server Manager.

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.

Identifying the Current Instance

When starting up the Oracle8 Enterprise Edition, you start up the current instance. The current Oracle8 instance is identified by the value of the logical name ORA_SID. For example, if the value of ORA_SID is currently V805, the current instance is the instance with the SID V805. If you have not reassigned the ORA_SID logical name, the value of ORA_SID is the SID specified during installation. To change the current instance before starting the Oracle8 Enterprise Edition with Server Manager, you should run the ORAUSER_<dbname>.COM file for the instance in question.

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

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

Specifying Startup Parameters

When the current Oracle8 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 Server Manager, 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:

```
SVRMGR> STARTUP PFILE=ORA_DB:INIT2.ORA
```

Starting the Server using Server Manager

To start Oracle8, you must have the process rights identifier ORA_DBA or ORA_<sid>_DBA assigned to your user account in the Alpha OpenVMS rights database and you must run the .COM file that makes the logical name assignments required to run Oracle8.

Before starting up Oracle8, run the ORAUSER_<dbname>.COM file to set the desired instance.

After running the above .COM file, run Server Manager and execute the appropriate STARTUP command(s), as documented in the *Oracle8 Server Administrator's Guide*. You can issue the single Server Manager command, STARTUP, or execute the three separate Server Manager commands documented in the *Oracle8 Server Administrator's Guide* to start the Oracle8 Enterprise Edition.

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

Starting Oracle8 Remotely via Server Manager from an OpenVMS Client

You can use Server Manager on an OpenVMS client to start up an Oracle8 database instance on a remote Alpha OpenVMS system.

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 ORA_DB:<nodename>_<sid>_INIT.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 ORA_DB:<nodename>_<sid>_INIT.ORA and ORA_DB:INIT.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:

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 `SQL*Net V8 ALIAS` for the remote system. For example:
3. Define the process logical name that points to the complete file specification for the `INIT` file copied in Step 6 above. For example:

```
$ DEFINE ORA_DFLT_HOSTSTR <SQL*Net V8 alias>
```

```
$ DEFINE ORA_PARAMS -
```

```
ddcn:[directory]<nodename>_<sid>_INIT.ORA
```

4. Edit the `<nodename>_<sid>_INIT.ORA`, `INIT.ORA`, and `INITPS.ORA` files and modify any `IFILE` parameters to point to the local directory on the client where these files are located.
5. Invoke `SVRMGR` 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.

```
$ svrmgrl
Oracle Server Manager Release 3.0.5.0.2 - Production
c) Copyright 1997, Oracle Corporation. All Rights Reserved.
Oracle8 Enterprise Edition Release 8.0.5.1.0 - Production
PL/SQL Release 8.0.5.1.0 - Production
SVRMGR> connect internal as sysdba
Password:
SVRMGR> startup
ORACLE instance started.
Total System Global Area          11381296 bytes
Fixed Size                        59952 bytes
Variable Size                     10969088 bytes
Database Buffers                   204800 bytes
Redo Buffers                       147456 bytes
Database mounted.
Database opened.
SVRMGR> exit
Server Manager complete.
```

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

Starting Oracle8 Remotely via Server Manager from a Windows PC Client

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>_PWFIL -  
ddcn:[directory]<fname>
```

If using a shared password file:

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

3. Edit ORA_DB:<nodename>_<sid>_INIT.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 ORA_DB:<nodename>_<sid>_INIT.ORA, ORA_DB:INIT.ORA, and ORA_DB:INITPS.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:

1. Ensure that there is a `TNSNAMES.ORA` entry for the `SID` on the remote system where the database resides.
2. Edit the `<nodename>_<sid>_INIT.ORA`, `INIT.ORA` files that were copied from the server and change all the `IFILE` parameters to point to the local DOS path to which these files were copied.
3. Invoke `SVRMGR` from File Manager (Windows 3.x) or Windows Explorer (Windows 95/98/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. `SQLNET_V8_ALIAS` is the `TNSNAMES.ORA` alias for the remote database.

```
Oracle Server Manager Release 3.0.5.0.2 - Production
(c) Copyright 1997, Oracle Corporation. All Rights Reserved.
Oracle8 Enterprise Edition Release 8.0.5.1.0 - Production
PL/SQL Release 8.0.5.1.0 - Production
SVRMGR> connect internal@<sqlnet_v8_alias>
Password:
SVRMGR> startup pfile=<DOS path to <node>_<sid>_INIT.ORA>
ORACLE instance started.

Total System Global Area          11381296 bytes
Fixed Size                        59952 bytes
Variable Size                    10969088 bytes
Database Buffers                  204800 bytes
Redo Buffers                      147456 bytes
Database mounted.
Database opened.
SVRMGR> exit
Server Manager complete.
```

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

Starting Oracle8 Automatically

To start Oracle8 automatically whenever you start Alpha OpenVMS, submit the Oracle8 start procedure as a batch job from the system startup file. This batch job must:

- Execute the `ORAUSER.COM` file to define the logical names and symbols referenced by Oracle8
- Run as the operating system DBA account user
- Run `ORA_RDBMS:INSORACLE.COM` to install the global sections required by Oracle8
- Execute one of the startup command files to start Oracle8:

```
$ @ORA_DB:STARTUP_EXCLUSIVE_<dbname>.COM
```

or

```
$ @ORA_DB:STARTUP_PARALLEL_<dbname>.COM
```

Sample Startup File

A sample startup file that starts two Oracle8 systems automatically after a system reboot is shown below:

```
$! STARTORAV8.COM
$! This script shows how one might start two Oracle
$! database instances at system boot time
$!-----$! Get the name of the node.
$!
$! NODENAME = F$GETSYI("NODENAME")
$!
$! Acquire CMKRNL privilege to install ORACLE
$! IMAGES. Exit with error if you are not so
$! authorized.
$!
$! SET PROCESS/PRIVILEGES=CMKRNL
$! IF (F$PRIVILEGE("CMKRNL").EQS. "FALSE") THEN EXIT 2
$!
$! Define symbols specific to this release of ORACLE
$! code by running the appropriate ORAUSER.COM:
$!
```

```

$ @DISK$ORACLE:[ORACLE.V8.UTIL]ORAUSER.COM
$!
$!   Install shared images:
$!
$ INSORACLE      ! Install shared ORACLE image
$!
$!   Start a database instance.
$!
$ INSTSID = "PROD1"      ! Define SID
$ DB_NAME = "PROD"      ! Define database name
$ GOSUB START_DATABASE
$!
$!   Start a second database instance.
$!
$ INSTSID = "PROD2"      ! Define sid
$ DB_NAME = "TEST"      ! Define database name
$ GOSUB START_DATABASE
$ EXIT
$!
$!   Invoke the database-specific startup script. Assumes
$!   that ORA_DB for each database is under ORA_ROOT.
$!   This need not be the case.
$!
$START_DATABASE:
$ @ORA_ROOT:[DB_'DB_NAME']STARTUP_EXCLUSIVE_'DB_NAME'.COM -
'INSTSID' 'NODENAME'
$ RETURN

```

In this sample startup file, the systems share the same copy of Oracle8 code. The example assumes that the Oracle8 root directory is `DISK$ORACLE:[ORACLE.V8]`.

Run this file as a batch job under the Oracle8 account as part of the standard system startup procedure. Keep this file in the Oracle8 account login directory.

For example, if the Oracle8 account resides in `DISK$ORACLE:[ORACLE]`, and the startup script is named `STARTORAV8.COM`, then you would start this script at boot time by adding the following lines to `SYS$MANAGER:SYSTARTUP.VMS_COM`:

```

$ filspc = "DISK$ORACLE:[ORACLE]STARTORAV8"
$ submit-
/user=Oracle8-
/after="+00:05:00"-

```

```
/log='filspc'.log-  
'filspc'
```

Shutting Down the Oracle8 Enterprise Edition

To shut down Oracle8, you can use one of these methods:

- Stopping Oracle Users Before Database Shutdown
- Shutting Down Oracle8 using ORACLEINS
- Shutting Down Oracle8 using the SHUTDOWN File
- Shutting Down Oracle8 using Server Manager

After all instances on a node have been shut down, you must de-install the shareable images. See the following section for information on de-installing images:

- De-installing Shareable Images

This section describes the three methods of shutting down Oracle8 and then tells how to deinstall shareable images.

Stopping Oracle Users Before Database Shutdown

SHUTDOWN IMMEDIATE will hang if you issue the command ALTER SYSTEM KILL SESSION <session> immediately followed by a HOST STOP/ID=<pid> on the processes associated with those Oracle sessions. 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. For example:

Process 1:

```
SVRMGR> startup
```


SVRMGR> select sid,serial#,process from v\$session;

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

6 rows selected.

Process 2 with OS-process id 20C00470:

\$ sqlplus <un>/<pw>

Process 1:

select sid,serial#,process from v\$session;

<u>SID</u>	<u>SERIAL#</u>	<u>PROCESS</u>
1	1	20C0018B
2	1	20C0018C
3	1	20C0018D
4	1	20C0018E
5	1	20C0018F
6	1	20C002DD
8	11	20C00470

7 rows selected.

SVRMGR> alter system kill session '8, 11';

Statement processed.

SVRMGR> host stop/id=20C00470

```
SVRMGR> shutdown immediate
```

```
... shutdown hangs ...
```

The solution is to use either ALTER SESSION KILL SESSION or HOST STOP/ID=. Don't use both. A pause before the SHUTDOWN so that PMON can clean up can also be a good idea.

Shutting Down Oracle8 using ORACLEINS

To shut down Oracle8 using ORACLEINS:

1. Using Server Manager, ensure that there are no open sessions.
2. Run the database-specific ORAUUSER file:

```
$ @ORA_DB:ORAUUSER_<dbname>.COM <sid> <setup_nodename>
```

3. Run ORACLEINS:

```
$ ORACLEINS
```

4. Select option 3, "Reconfigure existing products, manage the database, or load demo tables," from the ORACLE Installation Startup Menu.
5. Press [RETURN] when prompted to specify the root directory.
6. Press [RETURN] when prompted to specify the location of the savesets.
7. Select option 2, "Instance Creation, Startup, and Shutdown Menu" from the Main Menu.
8. Select option 4, "Shutdown an Existing Instance," from the Instance Creation, Startup, and Shutdown Menu. The following message is displayed:

```
Currently known database SIDs:
```

```
[list of known SIDs]
```

```
Press [RETURN] to quit with no action.
```

```
NOTE: The SID can be a maximum of 6 characters in length.
```

```
What is the SID for the instance you want to shut down?
```

9. Type the SID of the instance that you want to stop and press [RETURN]. The ORACLEINS utility will now do an orderly shutdown of the specified instance.

Shutting Down Oracle8 using the SHUTDOWN File

To shut down the currently running ORACLE instance, use the following command file:

```
$ @ORA_DB:SHUTDOWN_<dbname>.COM <sid> <setup_nodename>
```

This file is located in the database-specific directory identified by the logical name ORA_DB. When you shut down the instance, be sure to specify the SID of the instance and its setup node.

Sample Shutdown File

A sample shutdown file that shuts down two Oracle8 systems automatically is shown below:

```
$!
$!  NAME:  STOPORAV8.COM
$!  Note that this script will hang if users are still
$!  connected to the databases unless you modify the
$!  shutdown scripts to issue SHUTDOWN IMMEDIATE commands.
$!-----
$!
$!  Get the name of the node:
$!
$ NODENAME = F$GETSYI("NODENAME")
$!
$!  Acquire CMKRNL privilege to remove the Oracle
$!  shareable images. Exit with error if you are not so
$!  authorized.
$!
$ SET PROCESS/PRIVILEGES=CMKRNL
$ IF (F$PRIVILEGE("CMKRNL") .EQS. "FALSE") then exit 2
$!
$!  Define symbols and logicals specific to this release
$!  of the Oracle code by running ORAUSER.COM
$!
$!
$ @DISK$ORACLE:[ORACLE.V8.UTIL]ORAUSER.COM
$!
$!  Shut down a database instance
$!
$ INSTSID = "PROD1"                ! Define SID
$ DB_NAME = "PROD"                 ! Define Database Name
$ GOSUB DO_SHUTDOWN
```

```
$!  
$!   Shut down a second database instance  
$!  
$  INSTSID = "PROD2"           ! Define SID  
$  DB_NAME = "TEST"           ! Define Database Name  
$  GOSUB DO_SHUTDOWN  
$!  
$!  
$!   De-install Oracle shareables  
$!  
$  REMORACLE  
$  EXIT  
$!  
$  DO_SHUTDOWN:  
$  @ORA_ROOT:[DB_'DB_NAME']SHUTDOWN_'DB_NAME'.COM 'INSTSID'-  
  'NODENAME'  
$  RETURN
```

Shutting Down Oracle8 using Server Manager

You can shut down an instance of Oracle8 using Server Manager. See Chapter 5 in this manual for instructions on setting up Server Manager on your Alpha OpenVMS platform. Then, refer to the generic (platform-independent) Oracle Server documentation for instructions on using Server Manager.

De-installing Shareable Images

After shutting down all Oracle8 instances on a node, remove the shareable images by issuing the following command:

```
$ @ORA_RDBMS:REMORACLE.COM
```

For more information about these utilities, see Chapter 8 of this Guide.

Managing Instances

Because you access a database with an instance, managing instances is an integral part of database administration. This chapter describes how to control and manage both single and parallel instances.

The following topics are presented:

- Managing Single Instances
- Preparing an Instance for Parallel Query

Managing Single Instances

This section discusses the following topics:

- Components of Instances
- Controlling Instances
- Deleting Instances

Components of Instances

Information for each instance is recorded in a file named `ORA_RDBMS:ORA_RDBMS_SIDS.DAT`. This file contains the instance SID, the name of the database associated with the instance, and the name of the node where the instance was set up. This file is used to determine whether a SID is currently in use and if it is defined for a given database on a given node.

The following files are associated with an instance:

- Instance-specific initialization file
- Instance-specific trace file

Controlling Instances

You can use `ORACLEINS` to set up an instance on a particular node to access a particular database.

The following command files, located in the `ORA_DB` directory of each database, start up, shut down, and define logical names for all instances of database `<dbname>`:

```
STARTUP_EXCLUSIVE_<dbname>.COM  
SHUTDOWN_<dbname>.COM  
ORAUSER_<dbname>.COM
```

Deleting Instances

To delete an instance, perform the following procedure:

1. Shut down the instance on the node on which the instance is running.
2. Edit `ORA_RDBMS:ORA_RDBMS_SIDS.DAT` and remove the instance entry.
3. Edit the `ORA_UTIL.DATABASE.TXT` file and remove the instance entry.
4. Delete the `ORA_DB:<nodename>_<sid>_INIT.ORA` file.

5. Clean out the instance's trace files in ORA_DUMP.

Adding and Enabling Instance Threads

With Oracle8, each instance writes to its own log files. A given log file is not associated with any particular instance like rollback segments are, but is instead associated with a thread number.

Each instance that starts has the next available thread number assigned to it. Before a thread number can be assigned, it must have log files associated with it and these log files must be enabled. Refer to the *SQL Language Reference Manual* for more information.

Adding threads for new instances is accomplished in two steps:

1. Add log files to the database, associating each log file with a thread using the ALTER DATABASE command:

```
ALTER DATABASE ADD LOGFILE THREAD <number> <filespec> <storage>;
```

For example:

```
ALTER DATABASE ADD LOGFILE THREAD 2  
  'DISK$DEV1:[ORALOG]ORA_LOG3.RDO' SIZE 2000K REUSE,  
  'DISK$DEV1:[ORALOG]ORA_LOG4.RDO' SIZE 2000K REUSE;
```

When adding log files to your database, if you do not specify the thread number, the log file is associated with the thread number of the instance to which you are connected. Once a log file is associated with a thread, it is very difficult to change its thread number. Each thread you want to use must have at least two log files associated with it.

2. After log files are associated with a thread, you must enable the thread for it to be usable when you try to start your instances.

```
ALTER DATABASE ENABLE PUBLIC THREAD <number>
```

For example:

```
ALTER DATABASE ENABLE PUBLIC THREAD 4;
```

The thread for the initial instance is enabled by default, and all log files created with the initial database are associated with this thread. Additional instances, even if they are set up and have rollback segments identified in their

instance-specific INIT.ORA files, cannot be started until additional log files (associated with unique thread numbers) have been created and their respective threads have been enabled.

Preparing an Instance for Parallel Query

Parallel Query is an option that allows parallel query processing, index creation, and data loading.

To prevent a parallel query server from timing out, you can modify your INIT.ORA file by setting one of the following parameters:

- Set PARALLEL_SERVER_IDLE_TIME to 0
- Set PARALLEL_MIN_SERVERS to the same value as PARALLEL_MAX_SERVERS

These modifications will prevent trace files from being generated when the parallel query servers time out.

For more information on the parallel query option, refer to the *Oracle8 Server Administrator's Guide*.

Managing the Database

Ensuring that the Oracle8 Enterprise Edition operates successfully can involve tuning the system or modifying parameters. These tasks require a thorough understanding of Alpha OpenVMS system administration as well as the concepts documented in the Oracle8 Server Administrator's Guide.

This chapter presents the following topics:

- Creating Database Links
- Creating Multiple Control Files
- Managing Database Files
- Database Verification Utility and Other Useful Utilities
- Debugging Database Processes with ORAMBX

Server Manager and SQL*Net

When you start up Server Manager, a bequeath adapter connection will be made if no TNS connect description is supplied.

Additional Information: For more information about the bequeath adapter, please refer to the *SQL*Net for Alpha OpenVMS Configuration and User's Guide*.

On Alpha OpenVMS systems, Server Manager is linked two-task. This requires Server Manager to make connections to the Oracle8 Enterprise Edition using SQL*Net.

Starting Server Manager

After executing the ORAUUSER_<dbname>.COM file for a given instance, you can start using Server Manager.

SVRMGR Line-mode

When starting Server Manager in line mode, implement one of the following steps before invoking SVRMGR:

- Define the logical name ORA_DFLT_HOSTSTR to a valid SQL*NET V8 service name

or

- Use the bequeath adapter

To invoke Server Manager, enter the following command:

```
$ SVRMGRL
```

SET ECHO and SET TERMOUT Functionality in SVRMGR

In Server Manager (SVRMGR), the command SET TERMOUT OFF only stops output of SQL commands from going to the screen. The SET ECHO OFF command turns off echoing of the command being executed. SET TERMOUT OFF does not stop all output, just the output returned by a given SQL command.

Creating Database Links

If you are connected to a database, you can access data from another database via a database link, using SQL*Net.

The following is an example of a statement that creates a database link.

```
SQL> CREATE DATABASE LINK PROD CONNECT TO SCOTT
      2> IDENTIFIED BY TIGER
      3> USING '<SQL*Net connect string>';
```

<SQL*Net connect string> specifies a remote database and is defined in your SQL*Net TNSNAMES.ORA file or, if you are using Oracle Names, is known to Oracle Names. Please consult your SQL*Net documentation for more information.

If you have upgraded from Oracle V7.1.5 or below, and have database links that specify SQL*Net Version 1 connect strings, you must drop the links and recreate them with SQL*Net Release 8 connect strings.

After creating the link, you can query tables on the remote database by using the database link name. For example, to select data from the DEPT table of the database identified by database link PROD, you could enter the following command:

```
SQL> SELECT * FROM DEPT@PROD;
```

Creating Multiple Control Files

Two control files are created whenever you create a database. 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 *Oracle8 Server Administrator's Guide* for general information. Specific information for Alpha OpenVMS 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.

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:[DB_<dbname>].

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.

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 *Oracle8 Server Administrator's Guide*.

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.

CREATE TABLE

If you have export files generated before Version 6.0.31 with check constraints or defaults that span lines, the entire CHECK option condition statement (including the right parenthesis) must be on one line. If it is not, you will be able to create and export these tables, but not import them. An import attempt will cause an ORA-00921 error.

Correct:

```
CREATE TABLE TEST (  
    COL1 CHAR (10),  
    COL2 CHAR (1),  
    CHECK (COL2 IN ('1', '2', '3'))  
);
```

Incorrect:

```
CREATE TABLE TEST (  
    COL1 CHAR (10),  
    COL2 CHAR (1),
```

```
CHECK (COL2 IN  
( '1', '2', '3' ));
```

DROP TABLESPACE

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

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.

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:

```
SVRMGR> ALTER DATABASE RENAME FILE  
2> 'DISK$1:[ORACLE.TEST.DB_V8TEST]ORA_SYSTEM.DBS' TO  
3> 'MY$DISK:[ORACLE.TEST.DB_V8TEST]ORA_SYSTEM.DBS'  
SVRMGR> EXIT  
$ BACKUP/LOG/VERIFY -  
DISK$1:[ORACLE.TEST.DB_V8TEST]*.CON -  
MY$DISK:[ORACLE.TEST.DB_V8TEST]*.CON  
SVRMGR> EXIT
```

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):

```
$ SVRMGRL  
SVRMGR> CONNECT INTERNAL AS SYSDBA  
SVRMGR> SELECT * FROM V$DBFILES;  
SVRMGR> DISCONNECT
```

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.

```
$ SVRMGRL
SVRMGR> CONNECT INTERNAL AS SYSDBA
SVRMGR> SELECT * from V$DBFILES;
SVRMGR> SELECT * from V$LOGFILE;
SVRMGR> SHUTDOWN
SVRMGR> 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=(INTERLOCK,NOBACK)/DELETE /VERIFY -
<device>:[<dir>]<filename>.<ext> -
<new_device>:[<new_dir>]<new_filename>.<ext>
```

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

```
$ SVRMGRL
SVRMGR> CONNECT INTERNAL AS SYSDBA
SVRMGR> STARTUP EXCLUSIVE MOUNT <dbname>
```

5. Rename the file in the database using the exact string taken from DBA_DATA_FILES.

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

6. Back up the control files.

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


```
SQL> SELECT * FROM V$LOGFILE;
```
 - Look in the ORA_INSTANCE directory for the CREATE_<database>.SQL file. This file is created at install time and lists where the redo log files were created.
 - Dump the control file.
2. Shut down the database, make a second 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.

```
$ SVRMGRL
SVRMGR> CONNECT INTERNAL AS SYSDBA
SVRMGR> SHUTDOWN
SVRMGR> EXIT
$ BACKUP/IGNORE=(INTERLOCK,NOBACK) -
<old_device>:[<dir>]<filename>.<ext> -
<new_device>:[<new_dir>]<new_filename>.<ext>
$ SVRMGRL
SVRMGR> CONNECT INTERNAL AS SYSDBA
SVRMGR> 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 Server Manager, rename the files in the database using the ALTER DATABASE command. Specify the full file path.

```
SVRMGR> CONNECT INTERNAL AS SYSDBA
SVRMGR> 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 file must already exist. If either of these requirements are not met, the statement will fail.

4. Shut down the database using the following commands.

```
SVRMGR> SHUTDOWN
```

5. Back up the control files.
6. Restart the database using the following commands.

```
SVRMGR> CONNECT INTERNAL AS SYSDBA
SVRMGR> STARTUP OPEN <dbname>
SVRMGR> EXIT
```

Database Verification Utility and Other Useful Utilities

This section gives information about the following:

- Database verification utility
- Other useful utilities

Database Verification Utility

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

To use this utility to verify data in an Oracle8 release 8.0.5 database, point to the 8.0.5 files from your Oracle8 Enterprise Edition release 8.0.5 installation.

Additional Information: Refer to the *Oracle8 Server Utilities* manual. As this document mentions, Server Manager can also be used to verify your database.

Other Useful Utilities

The utilities listed in this section are included in the ORA_RDBMS directory, mostly for internal use. **USE AT YOUR OWN RISK.**

FILES

Given either the PID or process name, this command file uses SDA to show any files for which the process has open channels. You need the CMKRNL privilege to run this script.

```
$ FILES <PID_or_process_name> [<optional_selection _string>]
```

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.

The most common ORAMBX commands are:

DUMP	<level>	Dump call stack
PGA	<level>	Dump the fixed pga
SGA	<level>	Dump the fixed sga
SYSTEM	<level>	Perform system state dump
EVENT	<text>	Set process event
SESEVENT	<text>	Set session event

DUMP	<level>	Dump call stack
BLOCK	<dba><level>	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, these commands can only be issued to the server-process, which does all the work anyway. Running ORAMBX against the client application will result in an error from ORAMBX. This is normal.

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 major sections:

- Archiving Redo Log Files
- Backing Up the Database
- Exporting to and Importing from Multiple Tapes

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 *Oracle8 Server 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 Oracle8 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.

Two initial redo logs of 2000 Kb 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 *Oracle8 Server Administrator's Guide* for more information.

Note: When running in parallel 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.

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 Alpha OpenVMS BACKUP utility to copy them from disk to tape. You should never archive directly to tape. Refer to Compaq's document *VMS Guide to Tapes and Devices*, and the *Oracle8 Server 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.V8.DB_MIS]  
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.

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 parallel 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. To change only a specific instance, set the value of the LOG_ARCHIVE_START parameter to TRUE in the instance-specific INIT.ORA file. To change all instances, make the change in INIT.ORA itself.

3. Specify the destination of the archived files with the LOG_ARCHIVE_DEST parameter in the same parameter file (either the instance-specific INIT.ORA file, or INIT.ORA itself).
4. Restart the instance.
5. If the database is mounted in parallel mode, and you want other instances to archive automatically, repeat the steps above, skipping step2, if you added the LOG_ARCHIVE_START and LOG_ARCHIVE_DEST parameters to INIT.ORA (rather than the current instance's <setup_node>_<sid>_INIT.ORA parameter file).

You can also enable automatic archiving for a database instance that is running in ARCHIVELOG mode without changing INIT.ORA by using the Server Manager command ARCHIVE LOG as in the following command:

```
SVRMGR> 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 to a trace file in the ORA_DUMP directory.

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:

```
SVRMGR> 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:

```
SVRMGR> ARCHIVE LOG NEXT <destination>
```

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

```
SVRMGR> 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.

Backing Up the Database

A database backup is a block-by-block copy of the database files. If you are the DBA, you should make backups of the database regularly. You can do one of the following:

- 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 restore 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 Alpha OpenVMS utility BACKUP. The *Oracle8 Server 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.

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 ORA_SYSTEM.DBS and you are copying to a directory named ARCDIR you would enter the following:

```
$ BACKUP ORA_DB:ORA_SYSTEM.DBS -  
DISK$2:[ARCDIR]ORA_SYSTEM.DBS
```

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.

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 Server Manager, and enter the following command:


```
SVRMGR> 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 Server Manager and enter the following command:

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

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

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

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 SYSSERROR, not SYSSOUTPUT 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 restore the export file generated by the Export utility, use the Import utility. For information about using these utilities, refer to the *Oracle8 Server Utilities*.

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

Exporting to OpenVMS Machines

To export files to tape for transfer to an 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>
```

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.

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.

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.

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.

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.

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 chapter 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.

This chapter contains the following major sections:

- Overview
- Recovering from Instance Failure
- Recovering from Media Failure

Overview

Recovering an Oracle8 database is the process of restoring normal Oracle8 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 Oracle8 operation is to determine the type of failure that has occurred. There are four types of failure, but only two require any 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 *Oracle8 Server 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 *Oracle8 Server Administrator's Guide* and to the *Oracle8 Server Utilities* for information about the Oracle8 utilities used in recovery procedures.

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 Oracle8 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 CPUs, a failed instance will be recovered by one of the remaining functional instances within the cluster. You must still restart the failed instance, however.

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 prepared for media failure properly, you can restore both the system tablespace datafiles and the non-system tablespace datafiles.

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 *Oracle8 Server Administrator's Guide*.

Note: If you run in parallel 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.

Restoring from an Export File

Refer to the *Oracle8 Server Utilities* for information on how to restore 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 Server Manager utility before importing the export file.

1. Back up the current database, redo log, and control files with the OpenVMS BACKUP utility.
2. Edit the ORA_DB:CREATE_<dbname>.COM file and modify any parameters if desired (for example, increasing the initial data file's size).

3. CONNECT to Oracle8 as SYSTEM and run the CATDBSYN.SQL script from the ORA_RDBMS_ADMIN directory.
4. Create a second rollback segment in the SYSTEM tablespace. Refer to the *Oracle8 Server Administrator's Guide* for more information on creating rollback segments.

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;
```

Optimizing Oracle8

This chapter describes basic tuning activities that optimize the performance of the Oracle8 Enterprise Edition on Alpha OpenVMS. This chapter contains the following major sections:

- Tuning Memory Usage for the Oracle8 Enterprise Edition
- Installing Products in Shared Memory
- Reducing Database Fragmentation

Tuning Memory Usage for the Oracle8 Enterprise Edition

There are various ways of tuning your Oracle8 system to improve system performance. This section describes the following four ways:

- Adjusting INIT.ORA parameters
- Modifying Alpha OpenVMS process quotas
- Adjusting the SGA

Refer to the *Oracle8 Server Administrator's Guide* for more information about system tuning, space management, and database tuning.

Adjusting INIT.ORA Parameters

The most direct method of tuning the system is to adjust the startup parameters defined in the INIT.ORA file.

Modify startup parameters for any of the following reasons:

- To specify the name of the database to be used
- To improve system performance
- To use database space more efficiently
- To establish backup and recovery procedures

The ORACLEINS procedure creates a copy of the INIT.ORA file in the ORA_DB directory. Modify the copy of the distributed INIT.ORA file to make changes that will affect all instances.

Modify the instance-specific INIT.ORA file to override settings in the ORA_DB:INIT.ORA files and to tune specific instances (for example, you can tune them for different hardware capabilities or different types of usage).

If you edit the INIT.ORA file, it is a good idea to make a copy of the distributed file to preserve the original parameter values. You can also add comments to the file so the parameters you have changed are marked with their original values.

Changes to INIT.ORA take effect only after you restart the instance. Therefore, if you create a new parameter file, you must shut down the current instance, reassign the ORA_PARAMS logical name, and start the instance to reference the new startup parameters.

Modifying Alpha OpenVMS Process Quotas

At instance startup time, Alpha OpenVMS process quota limits are set for the Oracle8 Enterprise Edition detached (background) processes. Some of the background processes are ARCH, DBWR, LCK0-9, LGWR, PMON, RECO, SMON, the dispatchers (D<xxx>), and the multi-threaded server processes (S<xxx>).

How the Oracle8 Enterprise Edition Sets Process Quotas

The Oracle8 Enterprise Edition automatically sets and adjusts the Alpha OpenVMS process quota limits. Therefore, Oracle Corporation does not recommend that you define process quota logical names for the background processes (as you did with Oracle Server version 6) because they override the quotas set by the Oracle Server.

To see how the Server determines the values of these quotas, see Table A-1, "Oracle quota limits" in Appendix A of this guide.

Warnings about Modifying the Process Quotas

If you do need to increase the Alpha OpenVMS quota limits, you can use the Oracle8 process quota logical names to do so. If you plan to change a process quota other than ENQLM, first consult Oracle Support Services. If you have insufficient Alpha OpenVMS quota limits, you will receive the Oracle8 errors ORA-07623 and/or ORA-00445.

How to Change the Process Quotas

To change the process quotas for a particular instance of ORACLE, complete the following steps:

1. Log in to the Oracle8 account (or the account from which you will restart the instance).
2. Shut down the instance if it is currently running.
3. Define the Alpha OpenVMS logical name that sets a new quota for a background process associated with the instance.

You can either set the quota limit for a specific detached process or set the quota limit globally for all detached processes. The quota limit set for a specific detached process has precedence over quota limits set globally for all detached processes.

- To set the quota limit for a specific detached process, use the following logical name:

```
ORA_<sid>_<process>_PQL$_<quota logical name>
```

For example, to increase the WSQUOTA to 4096 for the background process, PMON, where the SID for a particular database is PROD, use the following command:

```
$ DEFINE/SYSTEM ORA_PROD_PMON_PQL$_WSQUOTA 4096
```

- To set the quota limit globally for all detached processes, use the following logical name:

```
ORA_<sid>_PQL$_<quota logical name>
```

For example, to increase the WSQUOTA to 4096 for all of the background processes where the SID for a particular database is PROD, use the following command:

```
$ DEFINE/SYSTEM ORA_PROD_PQL$_WSQUOTA 4096
```

If neither logical name is defined, Oracle8 will find a value based on the size of the SGA and other factors.

4. Start the instance to make the new quotas take effect.

Adjusting the SGA

The System Global Area (SGA) is an area of shared memory. It includes database block buffers, the redo log buffer, and the data dictionary caches. The SGA is accessed by user processes and background processes.

In Alpha OpenVMS, the SGA is implemented as a global section. This section is created at instance startup. It is mapped by background processes at startup. The SGA does not page out to a system paging file.

By default, the SGA is created as a non-file backed memory resident global section that is not pageable. This results in significantly faster startup of processes that map the SGA.

The INIT.ORA parameter `VLM_BACKING_STORAGE_FILE` is provided. When this parameter is set to `TRUE`, a backing file is used for the SGA. This is provided in case there is some reason to allow the SGA to page. This parameter also disables the use of OpenVMS Fast I/O. For the best performance, leave this parameter set to the default value of `FALSE`.

If the SGA is pageable, it is paged to its own backing file, `ORA_INSTANCE:ORA_<sid>_SGA.ORA`, which is the size of the SGA.

You can adjust the SGA size by modifying your INIT.ORA parameters. If the SGA size increases, you might need to reserve additional Alpha OpenVMS memory space for the SGA. The size of the SGA and the SGA buffers is displayed whenever you start an Oracle8 instance as in the following example:

Total System Global Area	6375984 bytes
Fixed Size	59952 bytes
Variable Size	5963776 bytes
Database Buffers	204800 bytes
Redo Buffers	147456 bytes

To show the size of the SGA at other times, use the following command, `SHOW SGA`:

```
SVRMGR> SHOW SGA
```

Installing Products in Shared Memory

If you have more than one concurrent Oracle product user, installing the product in shared memory can potentially save physical memory and increase performance.

Installing Oracle Products

You can also install some or all of the Oracle products in shared memory by running the `ORA_INSUTL.COM` file. If a product is installed into shared memory, each product needs additional global pages (these numbers are approximate) as indicated in Table 8-2:

Program	Alpha Pagelets
Export	10539
Import	10137
SQL*Loader	10502
SQL*Plus	14849
Server Manager	10252
Pro*C	21606
Table 8-2 Additional Global Pages Required by Oracle Products	

Running ORA_INSUTL

The ORA_INSUTL.COM file can be run from the ORA_INSTALL directory any time after the products have been installed. Run ORA_INSUTL as often as you want to install different products.

1. To run the ORA_INSUTL file, enter the following command:

```
$ @ORA_INSTALL:ORA_INSUTL
```

A list of all the Oracle products installed on your system is displayed, similar to the following:

```
Installable Utilities
-----
1. SQL*Plus
2. NetConfig
3. PROGINT
4. RDBMS
```

2. Enter the number of the product you want to install and press [RETURN]. Enter ALL to install all products. Enter E or EXIT to leave this menu with the products you selected. Enter Q for Quit to leave this menu without installing any product.

ORA_INSUTL creates ORA_UTIL:INSUTILITY.COM and ORA_UTIL:REMUTILITY.COM. It then runs INSUTILITY to install the selected products in shared memory and exits.

ORA_UTIL:INSUTILITY.COM and ORA_UTIL:REMUTILITY.COM are defined as follows:

- INSUTILITY.COM
 - Installs Oracle products other than the Server (RDBMS), such as Oracle Forms and SQL*Plus, in shared memory.
- REMUTILITY.COM
 - Removes the products from shared memory, such as Oracle Forms and SQL*Plus, installed by INSUTILITY.COM. It does not affect the shared global sections of the Oracle8 images.

Reducing Database Fragmentation

This section supplements the instructions for reducing database fragmentation in the *Oracle8 Server Utilities User's Guide*. Refer to that document for more information about the Export and Import utilities. Every time a structural change is made to the database, such as adding, moving, or dropping database files, back up the control files using the ALTER DATABASE command.

1. Shut down all instances associated with the database.
2. Start up the database so that it can be accessed only by DBAs as in the following example:

```
SVRMGR> STARTUP RESTRICT OPEN <dbname>
```

3. Perform a full database export (FULL=Y) to backup the entire database.
4. Use the MONITOR command in Server Manager to check for active users and shut down Oracle8 when all users are logged off.
5. Edit the ORA_DB:CREATE_<dbname>.SQL file and modify any parameters if desired (for example, when increasing the size of the initial datafiles).
6. Perform a full backup of your database.

Warning: Do not proceed to Step 7 until you have fully backed up your database.

7. Run the `ORA_DB:CREATE_<dbname>.COM` file to recreate your database.
8. CONNECT to Oracle8 as SYSTEM and run the `CATDBSYN.SQL` file from the `ORA_rdbms_admin` directory.
9. Next, create a rollback segment in the SYSTEM tablespace. Refer to the *Oracle8 Server Administrator's Guide* for instructions.
10. Add the name of the rollback segment to your `INIT.ORA` file and create any additional desired rollback segments.

You create private rollback segments using the following SQL statement:

```
CREATE ROLLBACK SEGMENT <name> <additional_parameters>
```

To take a private rollback segment in use, enter the following SQL command:

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

11. Import the export file using the Import utility.

Trace Files

This chapter describes how to use Oracle8 Enterprise Edition trace files when dealing with exception conditions.

This chapter contains the following major sections:

- Using Trace Files
- Specifying Trace File Directories
- Identifying Trace Files
- No Need to Format
- INIT.ORA Parameter for Creating World-readable Trace Files

Using Trace Files

Whenever Oracle8 encounters an exception condition, such as an access violation or an attempt to divide a value by zero, Oracle8 writes a trace file, also called a dump file.

A trace file can contain any of the following:

- Call stack trace
- Exception handler arguments
- Interpreted version of the exception handler data
- Register dump
- Dumps of the SGA, Process Global Area (PGA), and supervisor stacks
- Output from the SQL trace utility

The first few lines of the trace file include the time and date when the trace file was created and might contain other information about the creating process, including the following:

- Alpha OpenVMS process ID
- Alpha OpenVMS process name
- User identification code (UIC)
- User name
- Terminal device name
- Full filename specification of the image
- Process quotas and quota usage

Specifying Trace File Directories

Trace files are created by processes running the image ORACLE.EXE. These are the database processes, dedicated server processes, dispatchers and shared servers.

The INIT.ORA parameter, BACKGROUND_DUMP_DEST, sets the directory where trace files will be sent. Logical names can be used with this parameter rather than actual directory specifications. If the name is a logical name, then it is translated during instance startup in context of the process that starts up the instance.

Identifying Trace Files

The foreign command TRC is defined when the Oracle8 Enterprise Edition is installed. Use this symbol to display the trace files created in the ORA_DUMP directory on any given day.

Trace file names use the following convention:

`<nodename>_<sid>_<FG/BG>_<image>_<process_id>.TRC`

nodename	Name of the node the instance was running on when the trace file was created
sid	System ID of the instance that was running when the trace file was created
FG/BG	Indicates that the trace files were created during the execution of either a foreground process (FG) or background process (BG)
image	Name of the executable image that was running when the trace file was created
process_id	Three-digit ORACLE process ID that is the same as the ID that appears in the Server Manager Monitor screen.
.TRC	File name extension appended to all trace filenames

For example, a trace file created by process 005 running SQL*Plus against instance MKT1 might create a trace file called HARPO_MKT1_BG_ORACLE_005.TRC.

In addition to the above trace files, a file called `<nodename>_<sid>_ALERT.LOG` is stored in the background process dump directory and updated each time a number of different activities related to the database occur. You should be aware of the growth in size of the file over time. For more information about this file, see the README file.

In addition to the existing messages in `<nodename>_<sid>_ALERT.LOG`, the following messages result from the 64-bit feature:

- ** Reserve memory size = `<size>` greater than created SGA size = `<size>` **
 ** Please reduce reserved memory size to avoid wasting memory. **
- ** Memory was not reserved for the SGA. SGA size = `<size>` **
 ** There might be performance advantages to allocating memory for the SGA in the OpenVMS reserved memory registry. **

- **** Unable to create SGA Buffer Object - Fast I/O will not be used. non-fatal error = <error> ****

No Need to Format

The trace file format allows stack dumps to look similar on all implementations of Oracle8 Enterprise Edition and above. Trace files are also now preformatted.

INIT.ORA Parameter for Creating World-readable Trace Files

Trace files are created so that they are not world-readable. While this is secure, for those who are not administrating sensitive data the new protections may be overly restrictive. For example, a user who attempts to use SQL*Trace to analyze code behavior will find that the results are in a trace file that they cannot read.

When the internal INIT.ORA parameter `_TRACE_FILES_PUBLIC` is set to `TRUE`, trace files will be created world-readable. In this case, trace data is available and a user can use SQL*Trace to analyze code behavior. However, this is not a secure thing to do. Setting the parameter to `FALSE` is secure.

Part II

Using and Administering the Oracle Tools

Oracle Programmatic Interfaces

This chapter explains how to use the Oracle Programmatic Interfaces and SQL*Module in the Alpha OpenVMS environment. The following topics are covered in this chapter:

- Directory Structure
- Precompiling
- Compiling
- Linking
- Using the Oracle Call Interface (OCI) Routines
- Data Areas and Datatypes
- Using Literals as Call Arguments
- Optional or Missing Parameters
- Using Event Flags

Programmatic Interfaces

Warning: When upgrading from previous versions (7.1 or earlier,) you must recompile all your programmatic interface programs using the appropriate compilers.

The programmatic interfaces include:

- Pro*C
- Pro*COBOL
- Pro*FORTRAN

SQL*Module

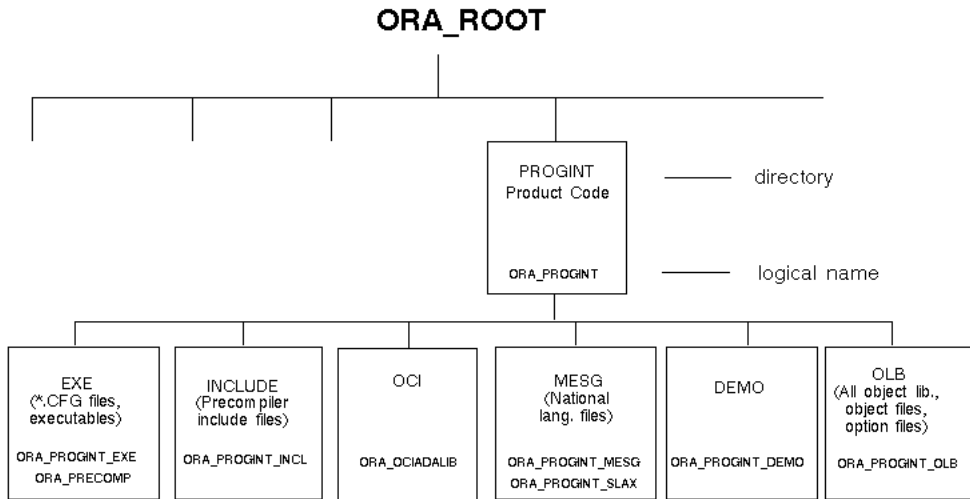
SQL*Module is a development tool that facilitates building and managing large applications that access data in an Oracle database. SQL*Module is available for the C language.

Additional Information: For general information, see the user's guide and README files for the programming language you are using.

Directory Structure

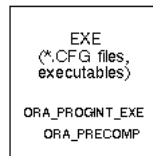
The following figure shows the directory structure created when the following Oracle Programmatic Interfaces are installed.

PROGINT Branch of the ORA_ROOT Directory Structure



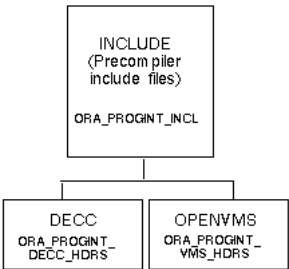
Precompiler Executable Files

The figure below shows the directory structure in the precompiler executable files.



Precompiler Include Files

The figure below shows the directory structure in the precompiler include files.



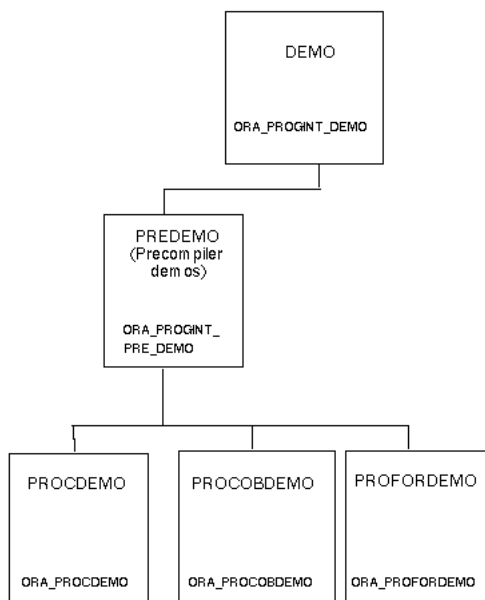
Precompiler Oracle Call Interface (OCI) Files

The figure below shows the directory structure in the precompiler OCI files.



Precompiler Demo Files

The figure below shows the directory structure in the precompiler demo files.



Precompiling

You invoke the precompilers and SQL*Module generator by using the OpenVMS symbols specified in Table 10-1.

Table 10–1 OpenVMS Symbols

Precompiler or SQL*Module	OpenVMS Symbol
Pro*C (release 8.0.5)	PROC
Pro*COBOL (release 1.8)	PROCOB
Pro*FORTRAN (release 1.8)	PROFOR
Pro*Cobol 8.0.5	PROCOB8

Syntax

Use the following syntax to precompile source files:

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

where:

<VMS_symbol>

OpenVMS symbol for the precompiler or SQL*Module.

C

C

<filename>

Name of the source file you want to precompile.

<option>

Precompiling option available for the Oracle Precompilers program. You can supply any number of option-value pairs, separated by a space.

<value>

Value for the option specified.

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.

Guidelines and Restrictions

The following guidelines and restrictions apply to precompiling.

Using the Alpha OpenVMS Debugger

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

Using Event Flags

If you use Alpha 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.

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.

Compiling

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

Compiler Options Used to Compile Oracle8

Oracle8 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 5.76 compiler on Alpha OpenVMS, the compilation options are as follows for most modules, with the exception of modules that can not be compiled with /OPTIMIZE:

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

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

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

If you compile your code with `/debug=trace`, line numbers in your modules will appear, as appropriate, in Oracle8 stack trace listings.

Floating Point Formats

Oracle8 is compiled with the default floating point format supported by the C compiler. The conversion routines within Oracle8 translate operating system-specific floating point numbers into Oracle8 internal floating point representation.

Note: Oracle precompiler and OCI programmers should take special note that Oracle8 Alpha OpenVMS is compiled to recognize the `G_FLOAT` floating point format.

Pro*COBOL

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

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

Note: All Oracle third party and user tools must now link against Oracle with the LOU TL "T" or "Z" option and connect to the database over SQL*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 SQL*Net TCP/IP adapter.

Syntax

To link compiled PRO<language> object files, use the LNPRO<language> symbol:

<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: DLinks with the Alpha OpenVMS DEBUG utility. FProduces a full map. MCreates a link map. XProduces a link map with cross references.

Note: The options S and T have been dropped for this release. Previously written scripts will still work but you should not include them in any new scripts.

Example

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

```
$ LNPROC MYFILE MYOBJ,SUB DM
```

Including Option Files

When you are using the Alpha 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,SYSS$INPUT/OPT
```

while the following command would work:

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

where MYOPT is the options file.

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.

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.

For more information on linking user exits, see the *Developer/2000 for OpenVMS User's Guide*.

Compatibility with ANSI Standard Compilers

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

Linking Shareable Images with LOU TL.COM

You may link a shareable image against Oracle8 code using the LOU TL I and D options with LOU TL.COM or one of the LNPRO*.COM link scripts that internally calls LOU TL.COM. To eliminate missing file errors, also use the LOU TL 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 LOU TL 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)

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.

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.

Using the Oracle Call Interface (OCI) Routines

OCI 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 Alpha OpenVMS for OCI programs.

OCI sample programs are supplied in the following directory:

```
ORA_ROOT:[RDEMS.DEMO.OCI_DEMO]
```

Guidelines

The following guidelines apply to using OCI routines:

- You can run OCI programs with the Alpha 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.

CDA/LDA Structure Information

For the C OCI programmer, the CDA and LDA structures (64 bytes each) are declared in the header file `ORA_ROOT:[RDBMS.DEMO.OCI_DEMO]OCIDFN.H`.

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

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 10-2

Table 10–2 Alpha Size and Offsets of Structure Elements

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

Linking OCI Programs Written in C

LNOCIC.COM is used to link OCI routines written in C. The syntax is:

```
$ @ORA_RDBMS:LNOCIC <executable> <objfilelist> <options>
```

where:

<executable>

Name of the executable image to be created; a filename extension is not required.

<objectfilelist>

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.

<options>

List of options with no separators needed:

DLinks with the Alpha OpenVMS DEBUG utility.

FProduces a full map.

MCreates a link map.

XProduces a link map with cross references.

For example:

```
$ @ORA_RDBMS:LNOCIC SAMPLE OBJECT1 D
```

Linking OCI Programs Written in Other Languages

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

```
$ @ORA_RDBMS:LNOCI <executable> <objectfilelist> <options>
```

For example:

```
$ @ORA_RDBMS:LNOCI SAMPLE OBJECT1 D
```

If you are linking object files in a library, use the /LIB qualifier and do not include your main or calling program in a library. If your command line exceeds the OpenVMS limit of 256 characters, you can use an option file with the /OPT qualifier.

Note: The Old Style OCI (HLI) function calls are not supported with the Oracle8 Enterprise Edition.

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_PRECOMP: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_PRECOMP:LNPRO<language>.COM. If commenting out this line eliminates your link errors, you must run that user-written utility without utilizing shared core. Running without shared core requires approximately 1500 additional OpenVMS blocks.

Data Areas and Datatypes

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

Binary Integers

For Alpha OpenVMS, binary integers are 32 bits and short binary integers are 16 bits, as shown in Table 10-4.

Table 10-3 Usage of Binary and Short Binary Integers

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

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. Alpha 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)
```

Optional or Missing Parameters

In Alpha OpenVMS, 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 this:

```
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.
```

Using Event Flags

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

Oracle8 prevents asynchronous events from interfering with synchronous events by overwriting their event flags. This may increase the reliability of Oracle8 software on modern hardware, but it may introduce some problems for application programmers.

Oracle8 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 Oracle8. Event flags 1 and 5 are used by all synchronous events within Oracle8 and can also be used by application programmers. SYSSGETEF() is not used for these event flags.

SQL*Net version 2 also uses additional event flags, which it gets dynamically from SYSSGETEF() 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.

This chapter explains how to use SQL*Plus in the OpenVMS environment. For detailed information about SQL*Plus, see the SQL*Plus User's Guide and Reference.

The following topics are covered in this chapter:

- Warning Message When Invoking SQL*Plus
- Interrupting SQL*Plus
- Installing Help Tables
- Running System Commands
- Passing Parameters to SQL*Plus
- Passing Values from SQL*Plus
- Using Profile Files
- Using the OpenVMS Editor from SQL*Plus
- Converting SQL*Plus Output Files
- Interpreting Output from the SQL*Plus TIMING Command
- Exit Status within DCL

Warning Message When Invoking SQL*Plus

When invoking SQL*Plus, you might see the message:

```
Error accessing PRODUCT_USER_PROFILE
Warning: Product user profile information not loaded.
You may need to run PUPBLD.SQL as SYSTEM
```

This means that the table `PRODUCT_USER_PROFILE` does not exist. This table is used to provide additional security.

You can ignore the warning. However, if you want additional, product-level security, you must create the `PRODUCT_USER_PROFILE` table. To do this, run the file `ORA_SQLPLUS_DEMO:PUPBLD.SQL` from the Oracle `SYSTEM` account. For more information, see Appendix E in the *SQL*Plus User's Guide and Reference*.

Interrupting SQL*Plus

SQL*Plus has its own `[CONTROL]-C` handler. For example, you cannot disable `[CONTROL]-C` from the OpenVMS environment by typing:

```
SET NOCONTROL=Y
```

Note that `[CONTROL]-C` and `[CONTROL]-Y` work differently in SQL*Plus. Pressing `[CONTROL]-C` interrupts SQL*Plus. If records are being displayed when SQL*Plus is interrupted, `[CONTROL]-C` terminates the SQL statement currently executing, stops the display, and returns you to the SQL*Plus prompt. You can also use `[CONTROL]-Y` to exit SQL*Plus; however, this sequence requires process recovery, so you should not use it under normal conditions.

Installing Help Tables

The SQL*Plus Help tables are not installed as part of the installation procedure. Instead, you must install these tables manually by executing the following command:

```
$ @ORA_SQLPLUS:HELPIINS
```

Before running this procedure, you must have SQL*Loader installed.

The procedure prompts you for the password of `SYSTEM`. At the prompt, enter the current `SYSTEM` password. For example, if you have not changed passwords, enter `"MANAGER."`

Running System Commands

To run DCL commands from SQL*Plus, precede the DCL command with a dollar sign (or the word HOST). Terminate HOST commands with a semi-colon, otherwise you might encounter DCL syntax errors. For example, you can use either of the following DIR commands:

```
SQL> HOST DIR [.NOTES];  
SQL> $ DIR [.NOTES];
```

The dollar sign indicates that the rest of the line should be passed to OpenVMS as a DCL command. SQL*Plus attempts to create a subprocess that will execute the command. Your OpenVMS account must have the privileges or quotas required to create subprocesses. If you cannot create a subprocess, SQL*Plus returns control to your terminal.

If you are issuing a DCL command from within SQL*Plus or another product with a SQL interpreter, the command cannot be longer than 256 characters. If, however, you issue the same command directly from the DCL line, the limit may be higher depending on the number of parameters the command can take.

For example, the SUBMIT command has eight optional parameters; therefore, you can use up to 9x256 characters to issue the command from the DCL command line—256 characters for each command or parameter. If you are issuing the SUBMIT command from within SQL*Plus, you have a total of only 256 characters for the entire command (including parameters).

Passing Parameters to SQL*Plus

This section discusses the three ways you can pass parameters to SQL*Plus:

- From OpenVMS
- Interactively from a user
- From an input file

Passing Parameters to SQL*Plus from OpenVMS

To pass parameters to SQL*Plus from the DCL command line, list the values of the parameters after the `@<sqlplus_script_file>`. If you want to preserve case, put the parameter values in double quotes.

For example, to pass the values SAL, EMP, and Adams to a script file named TESPARG.SQL from the DCL command line, enter:

```
$ SQLPLUS <username>/<password> @TESPAR SAL EMP "Adams"
```

Within the TESPARG script file, &1, &2, and &3 correspond to the values passed in the command line. You could then have the following statement in the script file:

```
SELECT &1 FROM &2 WHERE ENAME = '&3'
```

The only limit to the number of parameters that can be passed to SQL*Plus is the size of the command line. For more information, look up the topic “Substitution variables” in the *SQL*Plus User's Guide and Reference*.

Note: You should avoid naming your script files with names of I/O logical names (for example, 'TT'), as SQL*Plus may try to run the I/O logical name rather than the script file. As a general rule, care should be taken when using logical names.

When a parameter contains spaces, passing it to SQL*Plus is somewhat tricky. Suppose you have the following SQL script called TEST.SQL:

```
SELECT '&1'
```

```
FROM dual;
```

Then, as an example, you could pass the string “This is a test” by entering either:

```
$ SQLPLUS SCOTT/TIGER @TEST "'This is a test.'"
```

or:

```
$ SQLPLUS SCOTT/TIGER @TEST "\"This is a test.\""
```

Passing Parameters to SQL*Plus Interactively from a User

For all variables that have not been defined or passed as arguments in the command line, SQL*Plus prompts for values from the user. The ACCEPT statement can be used to prompt the user for the values of variables. For more information, refer to “Writing Interactive Commands” in Chapter 3 of the *SQL*Plus User's Guide and Reference*.

Passing Parameters to SQL*Plus from an Input File

You can pass parameters dynamically to SQL*Plus from other programs and utilities. The parameters are passed from the DCL command procedure, via the command line, to SQL*Plus.

The following example is a DCL command procedure that reads values from an input data file and passes those values to SQL*Plus:

```
SELECT &1 FROM EMP WHERE SAL = &2 AND ENAME = '&3'
```

The input file INFILE.DAT contains the parameter values in a predetermined order, one value on each line. For example:

```
SAL
```

```
5000
```

```
ADAMS
```

The DCL command procedure, TOSQL.COM, opens INFILE.DAT and reads the values from INFILE.DAT, each into a different symbol, before invoking SQL*Plus.

The following example shows the DCL command procedure:

```
$ OPEN/READ INF INFILE.DAT
$ READ INF PAR1
$ READ INF PAR2
$ READ INF PAR3
$ CLOSE INF
$ SQLPLUS <username>/<password> @TOSQL 'PAR1' 'PAR2' - "'PAR3'"
$ . . .
```

Using the single quote around the arguments is necessary to substitute the value of the symbol (PAR1, PAR2, and PAR3). The order and number of parameters is predetermined and known by the command procedure. Note the special use of quotes around the third parameter; this ensures that for the corresponding value (ADAMS), uppercase will be preserved.

As an alternative, the parameters in INFILE.DAT could have been placed in one line. Then, a DCL lexical function could be used to extract and separate the parameters.

Passing Values from SQL*Plus

Besides passing parameters to SQL*Plus, you can also:

- Pass a single value from SQL*Plus to OpenVMS
- Pass multiple values from SQL*Plus to OpenVMS through a file.

Values must be numeric; you cannot pass strings. However, a numeric value can be either a constant or a variable.

Passing a Numeric Value

To pass a single numeric value from SQL*Plus back to DCL, use the EXIT facility of SQL*Plus.

Constant

To pass a constant value, place that value after the EXIT statement of SQL*Plus. For example, to return the value 66 to DCL, execute the following command:

```
SQL> EXIT 66
```

The value 66 is placed in the system symbol \$STATUS. To check for this value or to use this value from DCL, type:

```
$ IF $STATUS .EQ. 66 THEN GOTO SUB66
```

Variable

To pass a variable value back to DCL, use the NEW_VALUE function of SQL*Plus. For example, to display the message “You are underpaid” on the screen whenever the value of SAL from EMP is below 1000, follow these steps:

1. Add the following lines to the SQL*Plus script file:

```
COLUMN VARY NEW_VALUE SALVAL  
BREAK ON VARY  
SELECT SAL VARY FROM EMP WHERE ENAME = 'JAMES';  
EXIT SALVAL
```

2. Add the following lines to the DCL command procedure:

```
$ SET MESSAGE/NOFACILITY/NOSEVERITY/NOIDENTIFICATION/NOTEXT  
$ SET NOON  
$ SQLPLUS <username>/<password> @sqlfile  
$ IF $STATUS .LT. 1000 THEN -
```

```
WRITE SYS$OUTPUT "You are underpaid"
$ SET ON
$ SET MESSAGE/FACILITY/SEVERITY/IDENTIFICATION/TEXT
```

The SET MESSAGE command suppresses the display of the DCL message that corresponds to the value returned in \$STATUS. The SET NOON command prevents the command procedure from aborting if the SQL*Plus exit value is interpreted as an OpenVMS error status.

Passing Multiple Values

To pass more than one value from SQL*Plus to DCL, write the values from SQL*Plus to an ASCII file. Then, open and read that file from the DCL command procedure.

When you use the SQL*Plus commands EXIT and WHENEVER SQLERROR EXIT to pass variables back to the operating system, the variable is stored in the OpenVMS symbol \$STATUS.

Note: This symbol stores only the status of the most recent command (that is, the status of the SQLPLUS command itself).

Using Profile Files

If you plan to run SQL*Plus often, with the same parameters and options in effect, you can store these preferences in one or more profile files. Whenever you run SQL*Plus, SQL*Plus looks for these files and calls up a session using the preferences specified in the files.

Types of Files

You can have one or both of the following profile files:

GLOBAL.SQL

Preferences that apply to all users at your site

LOGIN.SQL

Contains only preferences for a particular user

Order of Execution

If both files exist for a particular user, SQL*Plus first executes GLOGIN.SQL. Then it executes LOGIN.SQL. Thus, the preferences in LOGIN.SQL either supplement or override the preferences in GLOGIN.SQL.

Creating the Files

To create the files and make SQL*Plus use them, you must do the following:

- Edit the sample files
- Tell SQL*Plus where to find the files you created

Editing the Sample Files

A sample GLOGIN.SQL is in the ORA_SQLPLUS directory. Edit this file to specify your site's preferences.

Running DEMOBLD.COM creates a sample LOGIN.SQL file and places it in the directory from which you ran DEMOBLD. For each user who needs his or her own preferences, place a copy of LOGIN.SQL in the user's login directory and edit it accordingly.

Telling SQL*Plus Where the Files Are

To tell SQL*Plus where the site-profile files are, define the following logical names:

ORA_DATA

Directory containing GLOGIN.SQL

ORA_PATH

Directory containing LOGIN.SQL, and any other SQL script other than GLOGIN.SQL

When SQL*Plus looks for LOGIN.SQL, or any other SQL script, it first looks in the current directory, and then looks in the directory specified by ORA_PATH. When SQL*Plus looks for GLOGIN.SQL, it only looks in ORA_DATA. So, unless you define ORA_DATA, SQL*Plus will not execute GLOGIN.SQL.

For example, if you keep GLOGIN.SQL in ORA_SQLPLUS, execute the following command:

```
$ DEFINE/SYSTEM/NOLOG ORA_DATA ORA_SQLPLUS
```

Then, to tell SQL*Plus to look for LOGIN.SQL in each user's login directory, execute the following command:

```
$ DEFINE /SYSTEM/NOLOG ORA_PATH SYS$LOGIN
```

Using the OpenVMS Editor from SQL*Plus

You can bypass the SQL*Plus editor and use an OpenVMS editor such as TPU from SQL*Plus. When the user exits the OpenVMS editor, SQL*Plus regains control. Your OpenVMS account must have the privileges or quotas required to create subprocesses.

To invoke the editor, type:

```
SQL> EDIT
```

or:

```
SQL> ED
```

The current SQL*Plus text buffer is placed in the edit buffer and is given the temporary name AFIEDT.BUF. If you invoke the editor with a filename argument, as in:

```
SQL> ED QUERY4.SQL
```

the named file is placed in the edit buffer.

The default editor for SQL*Plus is the editor invoked by the EDIT command, usually TPU. To change this default, include the following line in your LOGIN.SQL for SQL*Plus, or execute it as a command from SQL*Plus:

```
DEFINE _EDITOR = '<editor>'
```

where *<editor>* is the name of the editor you want to run. For example:

```
DEFINE _EDITOR = 'EDIT/EDT'
```

Converting SQL*Plus Output Files

By default, text files created by SQL*Plus's SAVE command have the extension .SQL, and spool files created by SQL*Plus's SPOOL command have the extension .LIS. These files are StreamLF type rather than VariableLength type. If other utilities are not compatible with StreamLF type files, you may want to convert your .LIS files to VariableLength type.

To convert .LIS files:

1. Invoke EDT to create a new file with the extension .TXT:

```
$ EDIT/EDT <filename>.TXT
```

2. Include the .LIS file:

```
* INCLUDE <filename>.LIS
```

3. Exit the editor:

```
* EXIT
```

Interpreting Output from the SQL*Plus TIMING Command

The SQL*Plus TIMING command analyzes the performance of SQL*Plus commands and command files by writing their execution time to a timing area. Use the TIMING command to start and stop performance analysis and to display the contents of the current timing area.

On OpenVMS, output from the timing command appears as shown below:

```
ELAPSED: 0.00:00:01.33
```

where:

```
ELAPSED
```

Elapsed execution time, in seconds

Exit Status within DCL

Note that the DECC runtime library will modify the exit status of "0". If you specify:

```
SQL> EXIT 0
```

Then the DCL \$STATUS symbol will actually receive the value "1." This occurs for portability reasons: on UNIX systems, "0" is a success indicator and it is translated to the VMS success status of "1."

For more information, see the section on "exit" in the *DEC C Runtime Library Functions and Macros* reference manual.

PartIII

Appendices

Logical Names and Parameters

This appendix provides information about Oracle8 logical names and utilities, and the default and recommended values for various initialization parameters. Refer to the *Oracle8 Server Administrator's Guide* for general information about all the initialization parameters.

This appendix contains the following major sections:

- Oracle8 Logical Names
- System-Dependent Initialization Parameters

Oracle8 Logical Names

This section describes some of the most important logical names.

During installation, ORACLEINS writes several logical names. These assignments will be referenced in the ORA_UTIL:ORAUSER.COM file that is referenced whenever you start up, upgrade, link, or relink Oracle8 or other Oracle products. In addition, the first time any product is installed, ORACLEINS adds a call to the product-specific file that makes symbolic and logical name assignments to ORAUSER.COM.

Where Logical Names Are Defined

The logical names that are described in this section are defined in the following command files:

- ORA_UTIL:ORAUSER.COM defines:
 - ORA_INSTALL
 - ORA_ROOT
 - ORA_UTIL
- ORA_ROOT:[RDBMS]RDBMSUSER.COM defines:
 - ORA_ERROR
 - ORA_MAP
 - ORA_OLB
 - ORA_PLS
 - ORA_RDBMS
 - ORA_SLAX
- ORA_DB:ORA_DB_<dbname>.COM defines:
 - ORA_CONTROL1
 - ORA_CONTROL2
 - ORA_DB
 - ORA_INITSQL
 - ORA_PARAMS
- ORA_INSTANCE:ORAUSER_<dbname>.COM defines:
 - ORA_DUMP
 - ORA_INSTANCE
 - ORA_SID
- ORA_ROOT:[<product>]<product>USER.COM defines:

ORA_<product>
ORA_<product>_DEMO

Note: Oracle Support Services does not support modification of any command file, symbol, or logical name used by the Oracle8 Enterprise Edition and application development tools, except in the following situations:

- The client was instructed to make the changes by an Oracle Worldwide Technical Support analyst.
 - The Oracle documentation explicitly requests or requires the modification to be made to the command file, symbol, or logical name.
-
-

Alphabetical Listing

This section describes the most important Oracle8 logical names. They are presented in alphabetical order.

ORA_CONTROLx

Identifies the location and name of the control file(s) associated with the database to which your environment currently points. ORA_CONTROLx is defined by the ORA_DB:ORA_DB_<dbname>.COM file.

ORA_DATA

Identifies the directory containing the GLOGIN.SQL script. The site administrator must manually define this logical name. See on page 11-9 for more information.

ORA_DB

Identifies the directory where database-specific files reside (this includes the command files to start and stop instances and possibly the database itself). This directory also contains the instance-specific INIT.ORA file for each instance you set up to access a particular database.

A subdirectory is created for each database you create using the ORACLEINS procedure either under ORA_ROOT or in some other location that you specify. The directory is then populated with database-specific and instance-specific files that are

automatically created by ORACLEINS. ORA_DB is defined by the ORA_DB:ORA_DB_<dbname>.COM file.

ORA_DUMP

Identifies the location of the directory where Oracle8 trace files from the current instance are written. By default, this is the [TRACE] subdirectory of the ORA_DB directory. ORA_DUMP is defined by the ORA_DB:ORAUSER_<dbname>.COM file.

ORA_ERROR

Identifies the Oracle8 error message file. ORA_ERROR is defined by the ORA_ROOT:[RDBMS]RDBMSUSER.COM file.

ORA_INITSQL

Identifies the location and fully qualified name of the SQL script that executes when an Oracle8 database is created. This script sets up the core of the Oracle8 data dictionary and should never be modified. ORA_INITSQL is defined by the ORA_DB:ORA_DB_<dbname>.COM file.

ORA_INSTALL

Identifies a subdirectory of ORA_ROOT that contains dependency and configuration information for the products distributed to your site, the product list file, and the installation command files. ORA_INSTALL is defined by the ORA_UTIL:ORAUSER.COM file.

ORA_INSTANCE

Identifies the location where files associated with the current instance reside. These files assign instance-specific logical names and are typically created in ORA_DB when an instance is set up. ORA_INSTANCE is defined by the ORA_DB:ORAUSER_<dbname>.COM file.

ORA_MAP

Identifies the directory (the default is ORA_RDBMS) where the link map files are created during linking operations. Symbol map files are used to format Oracle8 trace files. ORA_MAP is defined by the ORA_ROOT:[RDBMS]RDBMSUSER.COM file.

ORA_OLB

Identifies the ORA_RDBMS and ORA_UTIL directories as a searchlist that contains object libraries distributed with Oracle8. Change this logical name if you move these object libraries to a central location that contains other libraries that are linked with your programs. ORA_OLB is defined by the ORA_ROOT:[RDBMS]RDBMSUSER.COM file.

ORA_PARAMS

Identifies the location and name of the INIT.ORA file associated with the current instance. This file resides in the ORA_INSTANCE directory. The INIT.ORA file, referenced by ORA_PARAMS, is used as the default INIT.ORA unless you explicitly state otherwise during startup. ORA_PARAMS is defined by the ORA_DB:ORA_DB_<dbname>.COM file.

ORA_PATH

Specifies the directory containing LOGIN.SQL. See Chapter 11, "Telling SQL*Plus Where the Files Are" for more information.

ORA_PLS and ORA_PLSQL

Identifies the message files for PL/SQL

ORA_RDBMS

Identifies the directory where the Oracle8 Enterprise Edition is installed. This directory contains the Oracle8 Enterprise Edition object libraries and the command files that build and link the shareable Oracle8 images from these libraries during the installation procedure. The shareable image will reside in this directory. ORA_RDBMS is defined by the ORA_ROOT:[RDBMS]RDBMSUSER.COM file.

ORA_ROOT

Identifies the top-level directory of the Oracle8 system, usually a subdirectory of the Oracle8 account's login directory. ORA_ROOT is defined directly by the ORA_UTIL:ORAUSER.COM file.

ORA_SID

Identifies the current Oracle8 instance. The value of this logical name is commonly known as the SID. It is assigned during installation. The SID also identifies the

System Global Area and the detached processes associated with the current instance. ORA_ROOT is defined by the ORA_DB:ORAUSER_<dbname>.COM file.

ORA_SLAX

Identifies the message files for PL/SQL.

ORA_UTIL

Identifies the subdirectory of ORA_ROOT where the Oracle8 utilities, libraries, and data files referenced by many Oracle products reside. This directory also contains the ORAUSER.COM file that defines Oracle8 logical names and command symbols and is executed whenever Oracle8 is invoked. ORA_UTIL is defined directly by the ORA_UTIL:ORAUSER.COM file.

ORA_<product>

Is usually used to identify the subdirectory of ORA_ROOT containing <product>; for example, ORA_SQLPLUS, ORA_PROGINT, ORA_SVRMGR, and so on. These product-specific ORA logical names are defined by the ORA_ROOT:[<product>]<product>USER.COM files.

ORA_<product>_DEMO

Identifies the subdirectory of ORA_<product> that usually contains product demo data. These product-specific ORA logical names are defined by the ORA_ROOT:[<product>]<product>USER.COM files.

TNS_ADMIN

Identifies SQL*Net V8 administration directories.

Process Quota Logical Names

If you don't set quotas for Oracle's background processes, the Oracle8 Enterprise Edition uses its own formulas to determine how to set the quota logical names. Table A-1 shows the formula for each quota logical name, along with the minimum and maximum values that you can use if you are setting the logical names yourself

Table A-1 Oracle quota limits

Quota Logical Name	Minimum	Maximum	Current Formula
PQL\$_ASTLM	0	65536	max # of dispatcherconnections + max # instances * max #processes/instance + 64 and 64 larger than write batch size
PQL\$_BIOLM	0	1024	max # of dispatcher connections + 64
PQL\$_BYTLM	1024	1,048,576	default/256000
PQL\$_CPULM	0	0	default/0
PQL\$_DIOLM	0	2048	20 larger than write batch size
PQL\$_ENQLM	256	32767	default/512
PQL\$_FILLM	64	65535	max # of dispatcher connections + 64
PQL\$_JTQUOTA	0	0	default/0
PQL\$_PGFLQUOTA	20480	4,194,304	See below
PQL\$_PRCLM	0	20	default/10
PQL\$_TQELM	10	2048	Number of background processes plus 20
PQL\$_WSDEFAULT	2048	100000	default/2048
PQL\$_WSEXTENT	2048	1,048,576	See below
PQL\$_WSQUOTA	2048	1,048,576	See below

In Table A-1, note the following:

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	<div>If backing file used: $PAGE_TABLE_SIZE(SGA) + 4 * P0_IMAGE_SIZE + P0_DYNAMIC_SIZE + COMFORT_ZONE)) / 512 + 1$ Without backing file used: $4 * (P0_IMAGE_SIZE + P0_DYNAMIC_SIZE + COMFORT_ZONE)) / 512 + 1$</div>
PQL\$_WSQUOTA	<div>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: $PAGE_TABLE_SIZE(SGA_SIZE) + 4 * (PAGE_TABLE_SIZE(P0_IMAGE_SIZE) + .6 * (P0_DYNAMIC_SIZE) + COMFORT_ZONE)) / 512 + 1$</div>

For more information about modifying Oracle process quotas through logical names, see on page 8-3 of this Guide.

System-Dependent Initialization Parameters

All parameters require an equal sign (=). For example, DB_BLOCK_SIZE = 8192 is correct.

BACKGROUND_DUMP_DEST

Purpose	Identifies the directory where the trace files created by the detached Oracle8 processes are sent.
Recommended Value	The directory identified by the ORA_DUMP logical name.
Default Value	ORA_DUMP. By default, the ORA_DUMP logical name is assigned to the [.TRACE] subdirectory of ORA_DB.
Distributed Value	None

CONTROL_FILES

Purpose	Identifies the name and location of the database control files.
Recommended Value	The names of all control files created by the database that can be accessed by the current instance: ORA_CONTROL1, ORA_CONTROL2.
Default Value	ORA_CONTROL1
Distributed Value	ORA_CONTROL1, ORA_CONTROL2.

DB_BLOCK_SIZE

Purpose	Identifies size (in bytes) of Oracle8 database blocks and the database buffers in the SGA.
Recommended Value	A multiple of the Alpha OpenVMS I/O block size of 512 bytes. The maximum value is 32768.
Default Value	2048
Distributed Value	None

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 and is VERY DANGEROUS
Recommended Value	Any valid disk filename
Default Value	ORA_ARCHIVE:
Distributed Value	None

LOG_ARCHIVE_FORMAT

Purpose	<p>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.</p> <p>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 <i>Oracle8 Server Administrator’s Guide</i> for more information on these variables.</p>
Recommended Value	Any valid file name format.
Default Value	ARCH%T_%S.ARC
Distributed Value	None

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

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 Alpha OpenVMS, the minimum value is 5 MB.
Default Value	5 MB
Distributed Value	None

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

USER_DUMP_DEST

Purpose	Identifies the location to which trace files created by user processes are sent.
Recommended Value	The directory identified by the ORA_DUMP logical name.
Default Value	ORA_DUMP. By default, the ORA_DUMP logical name is assigned to the [.TRACE] subdirectory of ORA_DB.

Distributed Value	None
-------------------	------

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 Alpha OpenVMS Fast I/O, slow process startup, and in most cases reduce performance.
Default Value	FALSE
Distributed Value	None

Messages and Codes

This appendix lists the Oracle8 messages and codes that are specific to the Alpha OpenVMS environment. These messages and codes supplement those in the *Oracle8 Server Messages*. This chapter also contains messages and codes that are common in, yet not reserved specifically for, the Alpha OpenVMS environment.

The following are Alpha OpenVMS messages and generic messages that are particularly applicable to the Oracle8 Enterprise Edition on Alpha OpenVMS. All messages between 07500 and 07999 are Alpha OpenVMS operating system dependent. For more information on Oracle messages and codes, refer to the *Oracle8 Server Messages*.

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

Cause: This is an Alpha OpenVMS error message that occurs when you try to run an Oracle8 tool without installing the Oracle8 shareable image. The most likely image names that will be listed is ORA_CLIENT_<image id>.

Action: Install Oracle8 in shared memory before the instance is started by executing the following command files:

```
$ @ORA_RDBMS: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 INTERNAL 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 INTERNAL depend on:

- whether an ORA_<sid>_DBA identifier is in the Alpha 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 Alpha OpenVMS AUTHORIZE utility. The following cases identify process rights identifiers and your subsequent privileges:

- If the identifier, ORA_<sid_x>_DBA, exists in the Alpha OpenVMS 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 Alpha OpenVMS 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 Alpha OpenVMS 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-01092 Oracle instance terminated. Disconnection forced.

Cause: The instance this process was connected to was terminated abnormally (for example, from a shutdown abort). This process was forced to disconnect from the instance.

Action: When the instance has been restarted, retry action.

ORA-1731 circular view definition encountered

Cause: When trying to access the Server Manager account on Alpha OpenVMS Alpha OpenVMS, it is possible that the error might be caused by running CATALOG.SQL and MONITOR.SQL from the SYSTEM account.

Action: Run CATALOG.SQL and MONITOR.SQL from the SYS account. Also, refer to the *Oracle8 Server Messages* for more information on possible causes and actions for this error.

ORA-07500:scglaa: \$cantim unexpected return

Cause: Alpha OpenVMS system service \$CANTIM returned an unexpected value.

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

ORA-07501:scgtoa: \$deq unexpected return

Cause: Alpha OpenVMS system service \$DEQ returned an unexpected value.

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

ORA-07502:scgcmn: \$enq unexpected return

Cause: Alpha OpenVMS system service \$ENQ returned an unexpected value.

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

ORA-07503:scgcmn: \$setimr unexpected return

Cause: Alpha OpenVMS system service \$SETIMR returned an unexpected value.

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

ORA-07504:scgcmn: \$hiber unexpected return

Cause: Alpha OpenVMS system service \$HIBER returned an unexpected value.

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

ORA-07505:scggt: \$enq parent lock unexpected return

Cause: Alpha OpenVMS system service \$ENQ returned an unexpected value.

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

ORA-07506:scgrl: \$deq unexpected return

Cause: Alpha OpenVMS system service \$DEQ returned an unexpected value.

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

ORA-07507:scgcm: unexpected lock status condition

Cause: A global locking system service returned an unexpected value.

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

ORA-07508:scgfal: \$deq all unexpected return

Cause: Alpha OpenVMS system service \$DEQ returned an unexpected value.

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

ORA-07509:scgfal: \$deq parent lock unexpected return

Cause: Alpha OpenVMS system service \$DEQ returned an unexpected value.

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

ORA-07510:scgbrm: \$getlki unexpected return on lockid %s

Cause: The Alpha OpenVMS system service \$GETLKI returned an unexpected value.

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

ORA-07511:sscgctl: \$ENQ unexpected return for master termination lock

Cause: The Alpha OpenVMS system service, \$ENQ returned an unexpected value.

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

ORA-07512:sscgctl: \$ENQ unexpected return for client termination lock

Cause: The Alpha OpenVMS system service, \$ENQ returned an unexpected value.

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

ORA-07513:sscgctl: \$DEQ unexpected returned an unexpected value

Cause: The Alpha OpenVMS system service, \$DEQ returned an unexpected value.

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

ORA-07514:scgcan: \$DEQ unexpected return while canceling lock

Cause: The Alpha OpenVMS system service, \$DEQ returned an unexpected value.

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

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: Alpha OpenVMS system service \$OPEN failed.

Action: Check for a system error message and refer to Alpha 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-07518:sfccf: illegal file creation option

Cause: An illegal creation option (reuse, etc.) was sent to sfccf.

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

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: Alpha OpenVMS system service \$CREATE failed.

Action: Check for a system error message and refer to Alpha 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-07523:sfccf: \$connect error

Cause: Alpha OpenVMS system service \$CONNECT failed.

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

ORA-07524:sfccf: \$write (zero file) error

Cause: Alpha OpenVMS system service \$WRITE failed.

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

ORA-07525:sfccf: \$close error

Cause: Alpha OpenVMS system service \$CLOSE failed.

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

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-07528:sfccf: \$connect error

Cause: Alpha OpenVMS system service \$OPEN failed.

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

ORA-07529:sfifi: \$close error

Cause: Alpha OpenVMS system service \$CLOSE failed.

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

ORA-07530:sfofi: \$open error

Cause: Alpha OpenVMS system service \$OPEN failed.

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

ORA-07531:ssfccf: \$DISPLAY error

Cause: Alpha OpenVMS system service \$DISPLAY failed.

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

ORA-07532:sfcfi: \$dassgn error

Cause: Alpha OpenVMS system service \$DASSGN failed.

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

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-07534:scginq: \$getlki unexpected return on lockid %s

Cause: Alpha OpenVMS system service \$GETLKI returned an unexpected value.

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

ORA-07535:sfrfb: illegal logical block number

Cause: An attempt was made to read a block beyond the end of the file.

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

ORA-07536:sfrfb: \$qio(read) error

Cause: Alpha OpenVMS system service \$QIO failed.

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

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-07538:sfwrt: illegal logical block number

Cause: An attempt was made to write a block beyond the end of a file.

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

ORA-07539:sfwrt: \$qio (write) error

Cause: Alpha OpenVMS system service \$QIO failed.

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

ORA-07540:sfwfb: write completion ast error

Cause: An asynchronous disk write operation completed abnormally.

Action: Check for a system error message and refer to Alpha OpenVMS system documentation, or contact Oracle Support Services.

ORA-07541:sfifi: Cannot identify zero-length file.

Cause: Cannot use the specified file since it is zero-length.

Action: Examine each of the specified files to determine which one is zero-length and replace that with a file that is not zero-length.

ORA-07542:sfccf: Cannot create-reuse anything but top version of file

Cause: You tried to CREATE-REUSE an ORACLE data file with a specific version number. The specified version number was above or below that of the top version of the data file. This poses a security risk.

Action: Do not specify the version number of the file when CREATE-REUSEing ORACLE data files. Alternately, you should specify the version number of the existing top version of this file.

ORA-07543:sfrfb: Cannot read from zero-length file

Cause: Cannot read any logical blocks from the specified file since it is zero-length.

Action: Examine each of the specified files to determine which one is zero-length and replace that with a file that is not zero-length

ORA-07544:sfqio: asynchronous I/O not completed successfully

Cause: The asynchronous I/O being performed did not complete successfully.

Action: Examine the additional error messages and refer to Alpha OpenVMS system documentation.

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

Cause: Alpha OpenVMS 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: Alpha OpenVMS system service \$OPEN failed.

Action: Check for a system error message and refer to Alpha 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-07549:sftopn: \$OPEN failure

Cause: Alpha OpenVMS system service \$OPEN failed.

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

ORA-07550:sftopn: \$CONNECT failure

Cause: Alpha OpenVMS system service \$CONNECT failed.

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

ORA-07551:sftcls: \$CLOSE failure

Cause: Alpha OpenVMS system service \$CLOSE failed.

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

ORA-07552:sftget: \$GET failure

Cause: Alpha OpenVMS system service \$GET failed.

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

ORA-07553:sfofi: out of open files

Cause: The number of open files has exceeded a Alpha OpenVMS Oracle8 compile time limit.

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

ORA-07554:sfcopy: source & destination logical block sizes must match

Cause: The destination file supplied to SFCOPY has a different logical block size than the source file.

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

ORA-07555:sfquiast: illegal pending value

Cause: An internal inconsistency was found during an asynchronous disk I/O.

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

ORA-07556:sfotf: \$create error

Cause: Alpha OpenVMS system service \$CREATE failed.

Action: Examine system error message and refer to Alpha 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: Alpha OpenVMS system service \$CREATE failed

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

ORA-07559:sfdone: asynchronous I/O not completed successfully

Cause: The asynchronous I/O being performed did not complete successfully.

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

ORA-07560:sltn: \$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 ORA_SID that is exceptionally long or defined circularly. If none, report as a bug.

ORA-07561:szprv: \$IDTOASC failure

Cause: Alpha OpenVMS system service \$IDTOASC failed

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

ORA-07562:sldext: extension must be 3 characters

Cause: An extension was found but it is of improper length.

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

ORA-07563:sldext: \$PARSE failure

Cause: Alpha OpenVMS system service \$PARSE failed.

Action: Check for a system error message and refer to Alpha 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:sltext: \$SEARCH failure

Cause: Alpha OpenVMS system service \$SEARCH failed.

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

ORA-07568:slspool: \$OPEN failure

Cause: Alpha OpenVMS system service \$OPEN failed.

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

ORA-07569:slspool: \$CLOSE failure

Cause: Alpha OpenVMS system service \$CLOSE failed.

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

ORA-07570:szrfc: \$IDTOASC failure

Cause: Alpha OpenVMS system service \$IDTOASC failed.

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

ORA-07571:szrfc: \$FIND_HELD failure

Cause: Alpha OpenVMS system service \$IDTOASC failed

Action: Check for a system error message and refer to Alpha 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: Alpha OpenVMS system service LIB\$SPAWN failed.

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

ORA-07574:szrfc: \$GETUAI failure

Cause: Alpha OpenVMS system service \$GETUAI failed.

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

ORA-07576:sspexst: \$GETJPIW failure on process id %s

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07577:no such user in authorization file

Cause: An attempt was made to set an INTERNAL password (for either DBA or OPER privilege), but the corresponding Alpha OpenVMS account (either ORA_<sid>_DBA or ORA_<sid>_OPER) hasn't been created yet.

Action: Add a Alpha OpenVMS account for ORA_<sid>_DBA and/or ORA_<sid>_OPER before trying to set a password for them.

ORA-07578:szprv: \$FIND_HELD failure

Cause: Alpha OpenVMS system service \$FIND_HELD failed.

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

ORA-07579:spini: \$DCLEXH failure

Cause: Alpha OpenVMS system service \$DCLEXH failed.

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

ORA-07580:spstp: \$GETJPIW failure

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07581:spstp: cannot derive SID from unexpected process name

Cause: A background process did not have a name in correct form.

Action: If the job name was changed, restore it. Otherwise, this is an internal error. Verify that you can reproduce the error and contact Oracle Support Services.

ORA-07582:spstp: SID has illegal value

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

Action: Refer to the *Oracle8 for Alpha OpenVMS Installation Guide* and Chapter 4, "Managing Instances" of this guide for information on setting the SID.

ORA-07584:spdcr: invalid value for ORA_sid_(proc_)PQL\$_item

Cause: A logical name used to set a detached process quota value has an invalid value (probably non-numeric).

Action: Examine the values of these logical names, correct the one in error, and retry.

ORA-07585:spdcr: \$PARSE failure

Cause: Alpha OpenVMS system service \$PARSE failed.

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

ORA-07586:spdcr: \$SEARCH failure

Cause: Alpha OpenVMS system service \$SEARCH failed.

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

ORA-07587:spdcr: \$CREPRC failure

Cause: Alpha OpenVMS system service \$CREPRC failed.

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

ORA-07588:spdcr: \$GETJPIW get image name failure

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07589:spdde: system id not set

Cause: The logical name ORA_SID does not translate to a valid value.

Action: Check the value of ORA_SID in the process that gets the error, and correct the installation or command procedures that caused ORA_SID to be set incorrectly.

ORA-07590:spdde: \$DELPRC failure

Cause: Alpha OpenVMS system service \$DELPRC failed.

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

ORA-07591:spdde: \$GETJPIW failure

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07592:sspgrpv: error obtaining required privileges

Cause: While obtaining needed privileges, an error was returned from SYSS\$SETPRV.

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

ORA-07593:ssprpv: error release privileges

Cause: While releasing privileges, an error was returned from SYSS\$SETPRV.

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

ORA-07594:spiip: \$GETJPIW failed

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07595:sppid: \$GETJPIW failure

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07596:sptpa: \$GETJPIW failure

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07597:spguns: \$GETJPIW failure

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07598:spwat: \$SETIMR failure

Cause: Alpha OpenVMS system service \$SETIMR failed.

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

ORA-07599:spwat: \$HIBER failure

Cause: Alpha OpenVMS system service \$HIBER failed.

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

ORA-07600:spwat: \$CANTIM failure

Cause: Alpha OpenVMS system service \$CANTIM failed.

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

ORA-07601:spmguno: \$GETJPIW failure

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07602:spgto: \$GETJPIW failure

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07605:szprv: \$PARSE_ACL failure

Cause: Alpha OpenVMS system service \$PARSE_ACL failed.

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

ORA-07606:szprv: \$CHKPRO failure

Cause: Alpha OpenVMS system service \$CHKPRO failed.

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

ORA-07607:szaud: \$SENDOPR failure

Cause: Alpha OpenVMS system service \$SENDOPR failed.

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

ORA-07608:szprv: \$GETUAI failure

Cause: Alpha OpenVMS system service \$GETUAI failed.

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

ORA-07609:szprv: \$HASH_PASSWORD failure

Cause: Alpha OpenVMS system service \$HASH_PASSWORD failed.

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

ORA-07610:\$GETJPIW failed in retrieving the user's MAC privileges

Cause: Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07612:\$GETUAI failed in retrieving the user's clearance level

Cause: Alpha OpenVMS system service \$GETUAI failed.

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

ORA-07613:szlglt: \$ASCTOID failure

Cause: Alpha OpenVMS system service \$ASCTOID failed.

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

ORA-07618:\$IDTOASC failed translating a secrecy level

Cause: Alpha OpenVMS system service \$IDTOASC failed while looking up the string representation in the rights database of a secrecy level.

Action: Define the entry in the rights database which the binary label you specified references.

ORA-07619:\$IDTOASC failed translating an integrity level

Cause: Alpha OpenVMS system service \$IDTOASC failed while looking up the string representation in the rights database of an integrity level.

Action: Define the entry in the rights database which the binary label you specified references.

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, Alpha OpenVMS system service \$CREATE failed.

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

ORA-07623:smscre: \$CRMPSC failure

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

Action: Examine the system error message and refer to Alpha 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 8000 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 ORA_UTIL: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-07624:smsdes: \$DBGSLSC failure

Cause: While deleting the system global area (SGA), Alpha OpenVMS system service \$DBGSLSC failed.

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

ORA-07625:smsget: \$MGBLSC failure

Cause: While mapping the system global area (SGA) during logon, the Alpha OpenVMS system service \$MGBLSC failed. The usual reason is that Oracle8 has not been started up.

Action: Examine the system error message and refer to Alpha OpenVMS system documentation. Start up Oracle8 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, Alpha OpenVMS system service \$CRETVA failed.

Action: Examine the system error message and refer to Alpha 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-07629:smpall: \$EXPREG failure

Cause: While creating the program global area (PGA) during logon, Alpha OpenVMS system service \$EXPREG failed. This often happens when the virtual memory page count quota is exceeded.

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

ORA-07630:smpdal: \$DELTVA failure

Cause: While deleting the program global area (PGA) during logoff, Alpha OpenVMS system service \$DELTVA failed.

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

ORA-07631:smcacx: \$EXPREG failure

Cause: While creating or extending a context area, Alpha OpenVMS system service \$EXPREG failed. This often happens when the virtual memory page count quota is exceeded.

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

ORA-07632:smsrcx: \$DELTVA failure

Cause: While deleting a context area, Alpha OpenVMS system service \$DELTVA failed.

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

ORA-07633:smsdbp: illegal protection value

Cause: The buffer debug function was called with an illegal value. This is an internal error.

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

ORA-07634:smsdbp: \$CRETVA failure

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

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

ORA-07635:smsdbp: \$SETPRT failure

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

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

ORA-07636:smsdbp: \$MGBLSC failure

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

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

ORA-07637:smsdbp: buffer protect option not specified when sga created.

Cause: An attempt was made to change the buffer protect mode when the SGA was not created with buffer protect debug option. This is an internal error.

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-07642:smprtset: \$CMKRNL failure

Cause: While attempting to set the protection of a region of memory, an error was returned from the \$CMKRNL system service.

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

ORA-07643:smsalo: SMSVAR is invalid

Cause: This is an internal error.

Action: Verify that you can reproduce the error and contact Oracle Support Services and provide the with your INIT.ORA file.

ORA-07647:sszfck: \$OPEN failure

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

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

ORA-07650:sgtu: \$GETJPIW failure

Cause: While attempting to get the user's terminal device name during logon, Alpha OpenVMS system service \$GETJPIW failed.

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

ORA-07655:slsprom: \$STRNLOG failure

Cause: While attempting to translate SYSS\$INPUT during a prompt for a password, Alpha OpenVMS system service \$STRNLOG failed.

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

ORA-07656:slsprom: \$GETDVI failure

Cause: While attempting to get device characteristics during a prompt for a password, Alpha OpenVMS system service \$GETDVI failed.

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

ORA-07657:slsprom: \$ASSIGN failure

Cause: While prompting for a password, Alpha OpenVMS system service \$ASSIGN failed.

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

ORA-07658:slsprom: \$QIOW read failure

Cause: While prompting for a password, Alpha OpenVMS system service \$QIOW failed.

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

ORA-07665:ssrexhd: recursive exception encountered %s %s %s %s %s %s

Cause: An Alpha OpenVMS exception occurred while executing in the Oracle8 exception handler. The message includes the signal number, first and second signal arguments, and exception PC, PSL, and R0. This is an internal error.

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

ORA-07670:\$IDTOASC failed translating a secrecy category

Cause: Alpha OpenVMS system service \$IDTOASC failed while looking up the string representation in the rights database of a secrecy category.

Action: Define the entry in the rights database which the binary label you specified references.

ORA-07671:\$IDTOASC failed translating an integrity category

Cause: Alpha OpenVMS system service \$IDTOASC failed while looking up the string representation in the rights database of an integrity category.

Action: Define the entry in the rights database which the binary label you specified references.

ORA-07680:sou2os: another call to Oracle8 currently executing

Cause: A call to the Oracle8 shared image entry point occurred from within the shared image.

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

ORA-07681:sou2os: an error occurred while initializing Oracle8

Cause: While attempting to set up the dispatch vectors for the shared image, an error occurred.

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

ORA-07682:sou2os: set kernel dispatch fail err

Cause: During Oracle8 shared image entry, a dispatch to kernel mode failed.

Action: Make sure that your shared image is installed with the CMKRNL privilege, then contact Oracle Support Services.

ORA-07683:sou2os: \$SETPRV reset error

Cause: During an attempt to restore user privileges at Oracle8 shared image exit, Alpha OpenVMS system service \$SETPRV failed. This is an internal error.

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

ORA-07687:smscre: invalid value in vlm_sga_base_address

Cause: vlm_sga_base_address init.ora parameter has invalid value.

Action: Please provide the right value.

ORA-07688:smscre: \$CREATE_REGION_64 failure

Cause: Alpha OpenVMS system service \$CREATE_REGION_64 failed.

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

ORA-07689:smscre: \$CRMPSC_GFILE_64 failure

Cause: Alpha OpenVMS system service \$CRMPSC_GFILE_64 failed.

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

ORA-07690:smscre: \$CRMPSC_GDZRO_64 failure

Cause: Alpha OpenVMS system service \$CRMPSC_GDZRO_64 failed.

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

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

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

Action: Add ORA_SGA identifier to the system.

ORA-07692:ssmsget: \$MGBSLC_64 failure

Cause: Alpha OpenVMS system service \$MGBLSC_64 failed.

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

ORA-07693:ssmsget: \$DELTVA_64 failure

Cause: Alpha OpenVMS system service \$DELTVA_64 failed.

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

ORA-07694:ssmsget: \$CREATE_REGION_64 failure

Cause: Alpha OpenVMS system service \$CREATE_REGION_64 failed.

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

ORA-07695:smsfre: \$DELETE_BUFOBJ failure

Cause: Alpha OpenVMS system service \$DELETE_BUFOBJ failed.

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

ORA-07696:smsfre: \$DELETE_REGION_64 failure

Cause: Alpha OpenVMS system service \$DELETE_REGION_64 failed.

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

ORA-07697:smscre: \$GRANTID failure

Cause: Alpha OpenVMS system service \$GRANTID failed.

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

ORA-07698:smsget: \$GRANTID failure

Cause: Alpha OpenVMS system service \$GRANTID failed.

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

ORA-07700:soarch: interrupt received

Cause: An interrupt was received while archiving the logs.

Action: Retry operation.

ORA-07701:soatln: internal exception: output buffer too small

Cause: Overflow of buffer for parsing archive control text string.

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

ORA-07702:unrecognized device type in archive text

Cause: Unrecognized device type in archive text.

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

ORA-07703:error in archive text: need '/' after device type

Cause: The archive control text in the ARCHIVE command is invalid; the device type (to indicate a file or tape) must be followed by a '/'.

Action: Refer to the *Oracle8 Server Administrator's Guide* for the proper syntax of the text.

ORA-07704:error in archive text: need ':' after device name

Cause: The archive control text in the ARCHIVE command is invalid; the device name must be followed by a ':'.

Action: Refer to the *Oracle8 Server Administrator's Guide* for the proper syntax of the text.

ORA-07705:soaprs: device name buffer too small

Cause: The buffer supplied for the device name is too small. This is an internal error.

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

ORA-07706:error in archive text: need disk file name

Cause: The archive control text in the ARCHIVE command is invalid. The disk file name is missing.

Action: Refer to the *Oracle8 Server Administrator's Guide* for the proper syntax of the text.

ORA-07707:error in archive text: need tape label name

Cause: The archive control text in the ARCHIVE command is invalid. The tape label name is missing.

Action: Refer to the *Oracle8 Server Administrator's Guide* for the proper syntax of the text.

ORA-07708:sksaprs: tape label name buffer too small

Cause: Cause:The buffer supplied for the tape label is too small. This is an internal error.

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

ORA-07709:sksaprs: archiving to a remote host is not allowed

Cause: The user specified a remote disk for archiving via DECnet.

Action: Archive to a disk on the local host.

ORA-07710:sksaprs: file name buffer too small

Cause: The buffer supplied for the file name is too small. This is an internal error.

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

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-07713:sksamtd: SYS\$MOUNT failure

Cause: Alpha OpenVMS system service SYS\$MOUNT failed.

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

ORA-07715:sksadtd: SYS\$DISMNT failure

Cause: Alpha OpenVMS system service SYS\$DISMNT failed.

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

ORA-07716:sksachk: invalid device specification for ARCHIVE

Cause: Alpha OpenVMS system service SYSSGETDVI failed.

Action: Specify a valid device in ARCHIVE control string.

ORA-07717:sksaalo: error allocating memory

Cause: Alpha OpenVMS system service LIB\$GETVM failed.

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

ORA-07718:sksafre: error freeing memory

Cause: Alpha OpenVMS system service LIB\$FREE_VM failed.

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

ORA-07721:scgcm: not enough OS resource to obtain system enqueue

Cause: A call to sys\$enq returned an error indicating that the operating system lacked the resources necessary to create a lock. This is caused by the Alpha OpenVMS errors SSS_EXENQLM or SSS_INSFMEM.

Action: Free up some of the required resource to allow the creation of the required lock.

ORA-07740:slemop: incorrect handle size (programming error)

Cause: Structures used for reading error message files do not match.

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

ORA-07741:slemop: \$OPEN failure

Cause: Alpha OpenVMS system service \$OPEN failed.

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

ORA-07742:slemop: \$CONNECT failure

Cause: Alpha OpenVMS system service \$CONNECT failed.

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

ORA-07743:slemop: incorrect error file attributes

Cause: An error message file is of incorrect format.

Action: Unless an error file has been changed, contact Oracle Support Services.

ORA-07744:slemcl: invalid error message file handle

Cause: The seal in a passed-in handle does not match correct value.

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

ORA-07745:slemcl: \$CLOSE failure

Cause: Alpha OpenVMS system service \$CLOSE failed.

Action: Check system error and refer to Alpha OpenVMS system documentation.

ORA-07746:slemrd: invalid error message file handle

Cause: The seal in a passed in handle does not match correct value.

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

ORA-07747:slemrd: \$READ failure

Cause: Alpha OpenVMS system service \$READ failed.

Action: Check system error and refer to Alpha OpenVMS system documentation.

ORA-07750:slemcr: fopen failure

Cause: An attempt to create a message file failed. This is an internal error.

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

ORA-07751:slemcr: malloc failure

Cause: An attempt to allocate a cache for a newly created message file failed. This is an internal error.

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

ORA-07753:slemcf: fseek before write failure

Cause: An attempt to seek before writing a message file cache element failed. This is an internal error.

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

ORA-07754:slemcf: fwrite failure

Cause: An attempt to write a message file cache element failed. This is an internal error.

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

ORA-07755:slemcf: fseek before read failure

Cause: An attempt to seek before reading a message file cache element failed. This is an internal error.

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

ORA-07756:slemcf: fread failure

Cause: An attempt to read a message file cache element failed. This is an internal error.

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

ORA-07757:slemcc: invalid handle

Cause: The seal in a passed-in handle does not match the correct value. This is an internal error.

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

ORA-07758:slemcw: invalid handle

Cause: The seal in a passed-in handle does not match the correct value. This is an internal error.

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

ORA-07759:slemtr: invalid destination

Cause: The destination string provided to the function is too short.

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

ORA-07760:slemtr: \$open failure

Cause: The \$OPEN service failed.

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

ORA-07800:slbtpr: invalid number

Cause: An impossible request for binary to decimal conversion was made.

Action: This conversion cannot be performed.

ORA-07801:slbtpr: invalid exponent

Cause: An impossible request for binary to decimal conversion was made.

Action: This conversion cannot be performed.

ORA-07802:slbtpr: overflow while converting to packed decimal

Cause: An impossible request for binary to decimal conversion was made.

Action: This conversion cannot be performed.

ORA-07803:slpdtb: invalid packed decimal nibble

Cause: An impossible request for decimal to binary conversion was made.

Action: This conversion cannot be performed.

ORA-07804:slpdtb: number too large for supplied buffer

Cause: An impossible request for decimal to binary conversion was made.

Action: This conversion cannot be performed.

ORA-07820:sspscn: SYSSCRELNM failure

Cause: An error was returned from the SYSSCRELNM function.

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

ORA-07821:sspsdn: SYS\$DELLNM failure

Cause: An error was returned from the SYS\$DELLNM function

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

ORA-07823:sspsqr: \$QIO failure

Cause: An error was returned from \$QIO while trying to queue a read to the process dump mailbox.

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

ORA-07824:sspain: \$SETIMR failure

Cause: An error was returned from SYS\$SETIMR while trying to queue a process spin-watch timer.

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

ORA-07825:sspsck: #QIO failure at AST level

Cause: An error was returned from SYS\$QIO while trying to read the process dump mailbox.

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

ORA-07826:sspscm: SYS\$GETDVIW failure

Cause: An error was returned from SYS\$GETDVIW while trying to get information about the process dump mailbox.

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

ORA-07840:sllfop: LIB\$GET_VM failure

Cause: An error was returned from LIB\$GET_VM while attempting to allocate memory for an I/O vector.

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

ORA-07841:sllfop: SYS\$OPEN failure

Cause: An error was returned from SYS\$OPEN while attempting to open the data file for reading.

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

ORA-07842:sllfcl: SYS\$CLOSE failure

Cause: An error was returned from SYS\$CLOSE while attempting to close the input data file.

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

ORA-07843:sllfcl: LIB\$FREE_VM failure

Cause: An error was returned from LIB\$FREE_VM while attempting to free the memory for the I/O vector.

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

ORA-07844:sllfop: LIB\$GET_VM failure

Cause: An error was returned from LIB\$GET_VM while attempting to allocate memory for data and index buffers.

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

ORA-07845:sllfcl: LIB\$FREE_VM failure

Cause: An error was returned from LIB\$FREE_VM while attempting to free memory used data and index buffers.

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

ORA-07846:sllfop: # byte record too big for # byte user buffer

Cause: The longest record in the file will not fit into the largest data buffer that can be allocated.

Action: Modify the RMS file to have smaller records.

ORA-07847:sllfop: \$CONNECT failure

Cause: An error was returned by SYSS\$CONNECT while attempting to open the data file.

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

ORA-07848:sllfrb: \$GET failure

Cause: An error was returned by SYSS\$GET while attempting to read the data file.

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

ORA-07849:sllfsk: \$GET failure

Cause: An error was returned by SYSS\$GET while attempting to skip records in the input file.

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

ORA-07850:sllfop: bad option

Cause: You are using a bad option to loader. Fixed= is one legal option. Check documentation for others.

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

ORA-07860:osnsoi: error setting up interrupt handler

Cause: An error occurred while setting up the control interrupt.

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

ORA-07861:sfqio: cannot write to file opened in read-only mode

Cause: A write operation was attempted on a file opened in read-only mode.

Action: Do not open the file in read-only mode.

ORA-07862:sfqio: cannot expand the file

Cause: The size of a file could not be expanded

Action: Check for additional error messages and contact your customer support representative

ORA-07863:ssfduic: \$PARSE failure

Cause: The Alpha OpenVMS 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-07864:ssfduic: \$OPEN failure

Cause: The Alpha OpenVMS system service \$OPEN failed.

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

ORA-07865:ssfduic: \$CLOSE failure

Cause: The Alpha OpenVMS system service \$CLOSE failed.

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

Alpha OpenVMS Process Control

This appendix presents some useful tips about managing your Alpha OpenVMS 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 Compaq documentation contains additional information on these topics.

The following topics are covered in this appendix:

- Interrupting and Terminating Oracle Operations
- Running Oracle Programs as Detached Processes

Interrupting and Terminating Oracle Operations

This section explains the following:

- Cancelling without Aborting the Oracle Image
- Cancelling with the Option to Continue
- Disabling Control Keys

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 tools such as SQL*Plus or SVRMGRL).

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 Alpha 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 Alpha OpenVMS command executes because a partial shutdown of your Oracle session occurs.

Known Limitations in Client-Server Connections

Any client tool you are using is connected to Oracle through a client-server SQL*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 SQL*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 Alpha OpenVMS while engaged in a client-server connection to Oracle (connecting to Alpha OpenVMS 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.

Disabling Control Keys

To disable the control keys, enter the command:

```
$ SET TERM/PASTHRU
```

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:[ORACLE.DB_TEST]ORAUSER_TEST.COM  
$ RUN <myprog>.EXE
```

You might need to include certain process quotas to map DCL into the detached process’s virtual space. Refer to Compaq’s documentation for more information.

Alpha OpenVMS Supplement to the Generic Documentation Set

This appendix is designed to be used as a supplement to the following documents:

Oracle8 Server Administrator's Guide

Oracle8 Application Developer's Guide

Oracle8 Server Concepts

Oracle8 Server Migration Guide

Oracle8 Server Utilities

This Appendix is not intended for use on its own. It is a supplement, and chiefly contains references to specific topics covered in your Oracle8 for Alpha OpenVMS documentation set.

The following topics are covered in this chapter:

- Server Guides
- Utilities Guide

Server Guides

This section lists and explains the server guides.

Archiver Process (ARCH)

Background processes are created automatically when an instance is started.

Archiving Mode

For more information about log archiving, see Chapter 6, "Backing Up and Archiving Your Database"

enabling auto	Valid option on Alpha OpenVMS
initial mode	The value of the LOG_ARCHIVE_START parameter is FALSE by default, which means the archiver is off until this value is changed by editing the INIT.ORA file.
specific steps	For more information about archiving redo files, see Chapter 6, "Backing Up and Archiving Your Database"

Audit Trail

creation of views	On Alpha OpenVMS, the Oracle8 database and its data dictionary are created with the audit trail views defined by the CATAUDIT.SQL script.
O/S audits	Operating system audits are not currently used with Oracle8 for Alpha OpenVMS.
directing records	Operating system audits are not currently used with Oracle8 for Alpha OpenVMS. .

Authentication through OS

A user can be authenticated in many different ways. For example, you can use the OS_AUTHENT_PREFIX initialization parameter to set a prefix that is concatenated to the beginning of the account name when a user logs in ORACLE without a user name or password.

You can also limit user control over a database through the `ORA_DBA`, `ORA_<sid>_DBA`, `ORA_OPER`, `ORA_<sid>_OPER`, and `ORA_<sid>_<rolename>_<A/D>` rights identifiers.

To control access by remote clients, use `REMOTE_OS_AUTHENT` and `REMOTE_OS_ROLES`. If `REMOTE_OS_AUTHENT` is set to `TRUE`, then remote clients can perform OPSS\$ logons. If `REMOTE_OS_ROLES` is set to `TRUE`, then remote clients can use OS ROLES and roles that are identified externally. For security reasons, the default for both parameters is `FALSE`.

Background Processes

For more information about background processes, see Chapter 1, "Introduction to Oracle8 on Alpha OpenVMS".

ARCH	For more information about archiving mode, see "Archiver Process (ARCH)" and "Archiving Mode."
DBWR	Oracle8 on Alpha OpenVMS does not support multiple DBWR processes.
names	The prefix <code>ORA_<sid></code> is concatenated to the beginning of every background process. The <code><sid></code> is the value of the logical name <code>ORA_SID</code> . For example, if the value of the logical name <code>ORA_SID</code> is <code>MIS</code> , then the DBWR process is identified as <code>ORA_MIS_DBWR</code> .
creating	Background processes are created automatically when an instance is started. For more information about setting up server processes, see Chapter 3, "Starting Up and Shutting Down Oracle8".

Backups

A set of sample database backup scripts are included to aid database administrators in their duties. For more information, please see

ORA_DB_BACKUP_DEMO:README_BACKUP.DOC.

For more information about your backup options, see Chapter 6, "Backing Up and Archiving Your Database".

Client/Server Communication

For more information about communication links, see the SQL*Net documentation.

Clusters, Estimating Size

Use the following figures for calculating cluster size in the Oracle8 Enterprise Edition guides:

- Fixed header size = 86 bytes
- Variable transaction header = 24 (INITTRANS value for the table)
- Row header = 4 bytes per row of a clustered table

Configuring Oracle8

Alpha OpenVMS supports both multithreaded and dedicated server configurations.

Control Files

default name	The default names of the two control files are ORA_DB:ORA_CONTROL1.CON and ORA_DB:ORA_CONTROL2.CON. The logical names ORA_CONTROL1 and ORA_CONTROL2 point to the two control files, which are created at installation.
specifying names	You can make copies of the default control file using any legal Alpha OpenVMS file names, such as ORA_DB:ORA_CONTROL3.CON or DISK\$MIS:[BACKUP]ORA_CNTRL3_MIS.CON.
minimum size	The minimum size is 2500 blocks (1250 Kb) For more information about the Alpha OpenVMS block size, see the <i>Oracle8 for OpenVMS Installation Guide</i> .

specification Refer to the paragraph in this section on specifying names for information about filename specification.

Creating a Database During Installation

You must use ORACLEINS to request the creation of a new database.

Data Dictionary Creation

The CATALOG.SQL script is run for the default database during the installation process. If you create a database with the create database command, you must run CATALOG.SQL manually.

Data Files

The default value of the DATAFILE parameter of the CREATE DATABASE command is:

'ORA_DB:ORA_SYSTEM.DBS', SIZE 60M REUSE

The maximum size of each data file is 4095Mb.

Database

admin O/S account	On Alpha OpenVMS, the Oracle8 DBA must possess at least one of the ORA_DB, ORA_<sid>_DBA, ORA_OPER, or ORA_<sid>_OPER process rights identifiers, and have a UIC group number larger than the SYSGEN parameter MAXSYSGROUP.
background, server processes	For more information, see "Introduction to Oracle8 on Alpha OpenVMS," in Chapter 1 of this guide.
block size	The default value of the INIT.ORA parameter DB_BLOCK_SIZE is listed in the section on the initialization parameters below. The current value on your system can be found by entering the following command: SVRMGR> SHOW PARAMETERS
datafiles	The installation process creates one default database file, ORA_DB:ORA_SYSTEM.DBS. It is not necessary to create the database files before installation. For more information about adding datafiles, see Chapter 8, "Optimizing Oracle8" and the section on database and tablespace files in the <i>Oracle8 Server Concepts Manual</i> .
limit on connections	No information is currently available.

database limits	<p>For the equation given in the Oracle8 Enterprise Edition Oracle8 Enterprise Edition guides, use the following figures:</p> <p>minimum database file size = 5 MB for the first file.</p> <p>database files= 1022 or value of DB_FILES in the INIT.ORA parameter file or value of MAXDATAFILES in CREATE DATABASE.</p>
redo log files	Maximum number of files=255
connect string	Refer to the <i>SQL*Net Release 8.0.5 for Alpha OpenVMS Configuration and User's Guide</i> for information about database string specification.
SQL files to run at startup	Oracle8 for Alpha OpenVMS does not require you to run any SQL files at startup.

For specific information on the ADDRESS = portion of the connect descriptor, refer to the *SQL*Net Release 8.0.5 for Alpha OpenVMS Configuration and User's Guide*.

Files

locations	For more information about designating archive destination devices, see Chapter 6, "Backing Up and Archiving Your Database", of this guide.
names	<p>ORA_DB:INIT.ORA is the name of the parameter file shared by all instances of a given database.</p> <p>ORA_DB:<setup_node>_<sid>_INIT.ORA is the name of the parameter file used by each individual instance.</p> <p>Refer to Chapter 5, "Managing the Database", in this guide for more information on file names.</p>

Indexes

overhead of index blocks The overhead for an index block consists of KCBH, KTBBH, KTBIT, KDXLE, or KDXBR. To find out the overhead on your machine, enter the following query:

```
SQL> SELECT * FROM v$type_size
      2 WHERE type IN ('KCBH','KTBBH','KTBIT','KDXLE','KDXBR');
```

space required No information is currently available for these calculations.

Installing Products

Refer to the instructions in the *Oracle8 for Alpha OpenVMS Installation Guide* for complete information about installation.

Instance

check identifier A database instance is identified by the value of the logical name ORA_SID. Use the commands SET INSTANCE and SHOW INSTANCE to display the connect string used to connect to the remote instance.

initial default The logical name ORA_SID sets the value of the initial local default instance.

Use the SET INSTANCE command to change the value. You can return to the default value using the SET INSTANCE LOCAL command or by using SET INSTANCE without a qualifier.

startup Use the STARTUP command.

Latches

No Alpha OpenVMS-specific information is currently available.

Listener Process

For information on TNS listener processes, refer to the *SQL*Net for Alpha OpenVMS Configuration and User's Guide*.

Log Files

The default value of the LOGFILE parameter of the CREATE DATABASE command is:

```
'ORA_DB:ORA_LOG1.RDO' , 'ORA_DB:ORA_LOG2.RDO' SIZE 2000K REUSE
```

MAXLOGFILES, MAXLOGMEMBERS

When using ORACLEINS, the Alpha OpenVMS install procedure, the default value for MAXLOGFILES is 32 and the default value for

MAXLOGMEMBERS is 2. The maximum value of MAXLOGFILES is 255 and the maximum value of MAXLOGMEMBERS is 5.

Monitors

content of fields	There are no Alpha OpenVMS-specific elements in the contents of the monitor fields.
interrupt	No information is currently available on the interrupt mechanism.
running MONITOR.SQL	No Alpha OpenVMS-specific information is currently available.

Multiple Instances

Oracle8 for Alpha OpenVMS supports multiple instances.

Parameter Files

case sensitive	OpenVMS is not case sensitive.
creating and editing	The ORA_DB: INIT.ORA parameter file and ORA_DB:<setup_node>_<sid>_INIT.ORA file for each instance are created upon initial installation, and can be edited as text files on OpenVMS using a system editor.
format	Refer to the ORA_RDBMS:INIT.ORA file for the default INIT.ORA file.
global cache values	For more information about global cache values, see the <i>Oracle8 for Alpha OpenVMS Installation Guide</i> . Refer to the ORA_DB:INITPS.ORA file for suggested settings of these parameters.
location	The INIT.ORA, INITPS.ORA, and <setup_node>_<sid>_INIT.ORA files are located in the ORA_DB directory of every database.
names	You can use any legal OpenVMS filename with the PFILE option. The INIT.ORA file referenced throughout this guide refers to any of the various INIT.ORA files such as ORA_DB:INIT.ORA, ORA_DB:INITPS.ORA, and ORA_DB:<setup_node>_<sid>_INIT.ORA.
O/S parameters	Refer to the following section in the guide on initialization parameters for more information.
tuning	Refer to Chapter 8, "Optimizing Oracle8" and Chapter 9, "Trace Files" in this guide for more information on tuning.

Program Global Area (PGA)

No information is currently available on PGA parameters.

Protocol Adapters

For more information about SQL*Net Oracle Protocol Adapters, refer to the SQL*Net documentation.

Recovery

Refer to Chapter 6, "Backing Up and Archiving Your Database", in this guide for information about recovery procedures.

Redo Log Files

Refer to Chapter 6, "Backing Up and Archiving Your Database", in this guide for information about redo log files.

archived file name format	Refer to the chapter on backing up and recovering your database in this guide for information on the LOG_ARCHIVE_FORMAT and LOG_ARCHIVE_DEST parameters.
destination search path	This is the value of the ORACLE_PATH variable.
default location	The redo log files are located by default in ORA_DB.
default size	The default value of the LOG_SIZE variable is 2000 KB (4000 Alpha OpenVMS blocks).
limited by LOG_FILES	The default value of LOG_FILES is 255, which is also its maximum value.

Resource Costs

No information is currently available on setting resource costs.

Roles

For the role specification syntax given in the Oracle8 Enterprise Server Guides, <ID> refer to your SID (defined by the logical name ORA_SID).

Refer also to the sections in this appendix on authentication through OS and the database administration O/S account for information on how to limit roles via the operating system.

For more information about groups, see the *Oracle8 for Alpha OpenVMS Installation Guide*.

OSOPER and OSDBA	<p>On Alpha OpenVMS, the OSOPER role is controlled by the ORA_OPER and ORA_<sid>_OPER process rights identifiers. The OSDBA role is limited to users with the ORA_DBA and ORA_<sid>_DBA rights identifiers. As with ORA_DBA, if an ORA_<sid>_OPER rights identifier exists for the OSOPER role on a given database, then users with the ORA_OPER process rights identifier have insufficient privileges to perform the duties authorized by the OSOPER role.</p> <p>For more information about the ORA_DBA limitation, see the <i>Oracle8 for Alpha OpenVMS Installation Guide</i></p>
------------------	--

using the O/S for security OpenVMS verifies that you have the necessary rights identifiers for your role when you log in.

Script File Names

Script file names are not case dependent in OpenVMS.

Segments Extents

No Alpha OpenVMS-specific information is currently available on extents.

SQL Scripts, Names, and Locations

CATAUDIT.SQL	ORA_RDBMS_ADMIN
UTLCHAIN.SQL	ORA_RDBMS_ADMIN
DIT scripts	These scripts are not provided for Alpha OpenVMS.
UTLEXCPT.SQL	ORA_RDBMS_ADMIN
UTLLOCKT.SQL	ORA_RDBMS_ADMIN
to run initially	<p>If you create your database with the CREATE DATABASE command, you must run CATALOG.SQL, CATEXP.SQL, and ULVIEW.SQL after the CREATE DATABASE command completes.</p> <p>If you are also using the procedural option, you must run STANDARD.SQL, DBMSSNAP.SQL, DBMSSNAP.SQL, DBMSUTIL.SQL, DIO.SQL, and DBMSPIPE.SQL. All of these scripts are run for you automatically by the database creation and upgrade scripts provided in the Alpha OpenVMS Oracle8 installation scripts.</p>
STANDARD & DBMSSTDY.SQL	ORA_RDBMS_ADMIN
DBMSNAP.SQL	ORA_RDBMS_ADMIN

SQL*Net and Networking

For a thorough explanation of SQL*Net and networking, refer to the SQL*Net manuals.

choosing	For information on networking in your environment, refer to the SQL*Net manuals.
integrated into RDBMS	The RDBMS listener process has been merged with the SQL*Net V8 listener process. You access the SQL*Net V8 listener process when you use the Multi-Threaded Server configuration of Oracle8.
installing	Refer to the Oracle8 for Alpha OpenVMS Installation Guide for more information on installing SQL*Net.
O/S comm. software	Refer to the <i>Oracle8 for Alpha OpenVMS Installation Guide</i> for all software and hardware requirements.
specifying links	Refer to the <i>Oracle8 for Alpha OpenVMS Installation Guide</i> and the <i>SQL*Net Release 8.0.5 for Alpha OpenVMS Configuration and User's Guide</i> for host string specifications.

System Global Area (SGA)

The SGA is described in Chapter 1, "Introduction to Oracle8 on Alpha OpenVMS", in this guide.

Table Size, Estimating

Clustered	No Alpha OpenVMS-specific information is currently available.
non-clustered	No Alpha OpenVMS-specific information is currently available.

Tuning Information

Refer to Chapter 8, "Optimizing Oracle8", in this guide.

Trace Files

Refer to Chapter 9, "Trace Files", in this guide.

Transaction Entries, Space Required

No Alpha OpenVMS-specific information is currently available.

Initialization Parameters

Parameter Name	Default Value
BACKGROUND_DUMP_DEST	ORA_DUMP
COMMIT_POINT_STRENGTH	1
DB_BLOCK_CHECKSUM	FALSE
DB_BLOCK_SIZE	2048
DB_FILES	80
DB_FILE_MULTIBLOCK_READ_COUNT	8 (Range of 1-65536/DB_BLOCK_SIZE)
DB_FILE_SIMULTANEOUS_WRITES	4
DISTRIBUTED_TRANSACTIONS	16
INSTANCE_NUMBER	0
LOG_ARCHIVE_BUFFER_SIZE	127
LOG_ARCHIVE_BUFFERS	4
LOG_ARCHIVE_DEST	ORA_ARCHIVE:
LOG_ARCHIVE_FORMAT	ARCH%T_%S.ARC
LOG_BUFFER	8192(4xDB_BLOCK_SIZE)
LOG_CHECKPOINT_INTERVAL	10000
LOG_SMALL_ENTRY_MAX_SIZE	80
OS_AUTHENT_PREFIX	ops\$
PROCESSES	50
SHARED_POOL_SIZE	5000000 bytes
SORT_AREA_SIZE	65536
SORT_READ_FAC	20
SORT_SPACEMAP_SIZE	512
TEMPORARY_TABLE_LOCKS	60

TRANSACTIONS_PER_ROLLBACK_	11
SEGMENT	
USER_DUMP_DEST	ORA_DUMP

Utilities Guide

This section lists and explains information about the utilities guide.

Creating or Initializing a Database

No information is currently available.

Default Buffer Size

No information is currently available.

Default RECORDLENGTH

No information is currently available.

Error Messages

Refer to Appendix B, "Messages and Codes" of this guide for a complete list of messages for Oracle8 on Alpha OpenVMS.

Installing Export Views

Export views are automatically installed by the Oracle8 installation scripts. If they must defined at another time, run the script called

ORA_RDBMS_ADMIN:CATEXP.SQL.

Redirecting I/O

No information is currently available.

Backslash Escape

No information is currently available.

Sizes of Data

No information is currently available.

Size of SMALLINT Data Type

No information is currently available.

Length Subfield in VARCHAR Data

No information is currently available.

Length Subfield in VARGRAPHIC Data

No information is currently available.

File Processing Options

Current options available for Alpha OpenVMS are "FIXED=<n>", "STREAM=<n>", and "VAR" where n represents the number of bytes in the resulting file. You must put one of these options in double quotes (") on the command line.

For example, to declare a file in SQL*Loader named MYDATA.DAT as a file containing 80-byte records, use the following clause:

```
INFILE MYDATA.DAT "FIXED=80"
```

For example, to declare a file in SQL*Loader named MYDATA.DAT as a file containing variable length records, use the following clause:

```
INFILE MYDATA.DAT "VAR"
```

Examples of records in MYDATA.DAT that include first name, last name, and hire date are:

```
0025JOHN, SMITH, 09-JUNE-1981  
0032ROBERT, MILLER, 20-FEBRUARY-1976
```

Note that in Alpha OpenVMS you cannot specify the number of I/O buffers to be used in an operation

Index

A

Ada, 10-5
ALTER DATABASE command, 5-4, 6-2
ALTER SYSTEM command, 2-5
ALTER TABLESPACE command, 5-4
Appendices, 11
ARCH process, 6-4
ARCHIVELOG mode, 1-7
AUTHORIZE utility, 2-3

B

BACKGROUND_DUMP_DEST parameter, 9-2
Backup procedures, 1-7
Binary integers, 10-15
Blocksize, 5-4
Buffers
 database buffer pool, 1-5
 redo log buffers, 1-5

C

C Language
 precompiling, 10-5
CATDBSYN.SQL script, 7-5, 8-8
COBOL, 10-5, 10-16
Commands
 ALTER DATABASE, 5-4
 CONNECT INTERNAL, B-2
 CREATE TABLE, 5-4
 DROP TABLESPACE, 5-5
Compiling with programmatic interfaces, 10-7
CONNECT INTERNAL, B-2

CONTINUE command, C-2

Control files
 creating, 5-3
 identifying, 5-3

D

Database
 automatic recovery, 7-2
 granting access to, 2-3
 opening in exclusive mode, 3-4
Database files
 defined, 1-6
Database links
 creating, 5-3
Database verification utility, 5-8
database verification utility, 5-8
Datafiles, 1-8
Datatypes, 10-15
DB_BLOCK_BUFFERS parameter, 1-5
DB_dbname directory, 5-4
DCL command
 from within SQL*Plus or SQL*Forms, 11-3
DEBUG qualifier, 10-7
Demo files
 programmatic interfaces, 10-10
Detached processes, 7-3
 identification, A-6
DROP TABLESPACE command, 5-5

E

EDIT command, 11-9, 11-10
Editing files in SQL*Plus, 11-9

Errors

ORA-00445, 8-3

ORA-00921, 5-4

ORA-07623, 8-3

event flags

restrictions, 10-7

Exclusive mode, 3-4

Export utility, 6-8

for reducing fragmentation, 8-7

F

FORTRAN, 10-5, 10-16

Fragmentation, 8-7

H

Help

SQL*Plus Help tables, 11-2

HOST

DCL command, 11-3

option for PRO command, 10-6

I

Import utility

for recovery, 7-4

INCLUDE option, 10-6

INIT.ORA file

location, A-5

INIT.ORA files

at start-up, 1-4

setting SGA size, 1-4

INSORACLE.COM file

run as a batch job, 3-10

Interrupt keys

SQL*Plus, 11-2

I/O

improving, 1-5

J

Java, 1-2

JDBC drivers, 1-2

L

limit

on length of HOST command, 11-3

linker directives

with Oracle programming interfaces, 10-10

Linking

LNPRO symbol, 10-8

LOUTL.COM, 10-10

OCI routines, 10-13, 10-15

LNOCIC.COM, 10-13

LNOCI.COM, 10-14

LOG_BUFFER parameter, 1-6

LOG_CHECKPOINT_INTERVAL parameter, 1-6,
6-2

Logical names

alphabetical listing, A-3

command files that define, A-2

Logicals

identification, 1-8

using to specify files, 1-7

LOGIN.COM file

of ORACLE users, 2-2

LOGINOUT, C-3

LOGIN.SQL, 11-9

LRS

See Log roll-forward server (LRS), A-1, B-1, C-1,
D-1

M

Managing Instances, 4-1

N

NOARCHIVELOG mode, 1-7

O

OCI routines, 10-12, 10-15

options file

using for linker directives, 10-10

ORA_DATA logical name, 11-8

ORA_DUMP directory, 6-4

ORA_INSTALL logical name, A-4

ORA_INSUTL

- using to install shared images, 8-6
- ORA_PARAMS logical, 8-2
- ORA_PATH logical name, 11-8
- ORA_SID logical, 3-5
 - default value, 1-4
- ORA_USER.COM file, A-6
- ORA-00445 error, 8-3
- ORA-00921 error, 5-4
- ORA-07623 error, 8-3
- Oracle
 - contacting Worldwide Technical Support, xviii
- ORACLE call interface
 - OCI routines, 10-12
- ORACLE programmatic interfaces
 - compiling, 10-7
 - datatypes, 10-15
 - FORTTRAN %REF directive, 10-15
 - HOST option, 10-6
 - linking, 10-8
 - LNOCIC.COM, 10-13
 - LNOCI.COM, 10-14
 - LNPRO symbol, 10-8
 - OCI routines, 10-13, 10-15
 - optional call parameters, 10-16
 - precompiling, 10-5
 - PRO symbol, 10-7
- Oracle7
 - code, 1-3
 - granting access privileges, 2-3
 - linked as shared image, 1-4
 - logical names, A-2
 - relinking, 1-4
 - shutting down, 3-12
- ORACLE.EXE file
 - defined, 1-4
- ORACLEINS
 - to shut down Oracle7, 3-14
- ORACLEINS.COM file
 - logical assignments made, A-2
- ORAMBX, 5-9
- ORAUSER_instance.COM file, 2-2
 - running for ORACLE users, 2-2

P

- Parallel Query, 4-4
- Parameters
 - optional with programmatic interfaces, 10-16
 - passing to SQL*Plus, 11-3, 11-4, 11-5
- Pascal, 10-5
- password utility, 2-3
- Performance analysis, 11-10
- PL/I, 10-5
- Precompiling
 - guidelines and restrictions, 10-7
 - options, 10-6
 - syntax and example, 10-5
- preface
 - PT PrefaceTitle, xi
- Privileges
 - and Process Rights Identifier, B-1
- Process Global Area
 - dump files for, 9-2
- Process ID
 - OpenVMS, 9-2
- Process name
 - OpenVMS, 9-2
- Process Rights Identifier, B-1
- PRODUCT_USER_PROFILE, 11-2
- profile files
 - SQL*Plus
 - Pages, 11-7
- Programmatic Interfaces
 - ORACLE programmatic interfaces, 10-2
- PT PrefaceTitle, xi

Q

- Quotas
 - process quota logicals, 8-3

R

- Redo log file
 - mode of use, 6-2
- Redo log files
 - default names for, 1-7
 - defining, 1-6
 - modes of, 1-7

- moving, 5-7
- reusing, 1-7
- size of, 1-7
- Relinking Oracle7, 1-4
- Rolling backward, 7-2
- Rolling forward, 7-2
- Runtime libraries
 - used with Oracle7, 1-3

S

- Server Manager, 3-5
- Shared memory
 - installing products in, 8-5
- Shared ORACLE7 image, A-5
- Shared Pool, 1-5
- SHUTDOWN command, 7-3
- SHUTDOWN file
 - to stop ORACLE7, 3-15
- SID
 - and instances, 4-2
 - ORA_SID, A-5
- SPOOL command, 11-9
- SQL*Plus
 - editing files in, 11-9
 - GLOGIN.SQL, 11-7
 - help tables, 11-2
 - HOST command, 11-3
 - interrupt keys, 11-2
 - LOGIN.SQL, 11-7, 11-9
 - passing parameters to, 11-3, 11-4, 11-5
 - passing values from, 11-6, 11-7
 - PRODUCT_USER_PROFILE, 11-2
 - running system commands from, 11-3
 - SPOOL command, 11-9
 - TIMING command, 11-10
- STARTUP command, 3-5, 7-3
- System Global Area
 - components, 1-5
 - defined, 1-4
 - dump files for, 9-2
 - identification, A-6
 - size, 1-4
- SYSTEM tablespace, 7-4
 - removing, 5-4

T

- Tablespace files
 - moving, 5-6
- TIMING command, 11-10
- Trace files, A-4
- tracing
 - and SQL trace utility, 9-2

U

- User Identification Code (UIC), 9-2
- Utility
 - database verification, 5-8
 - useful, 5-8

V

- VMS Mailbox driver
 - creating database links for the, 5-3