Oracle7 for OpenVMS Server and Tools Administrator’s Guide

Release 7.3.4
About this Guide

The Oracle7 for OpenVMS Server and Tools Administrator’s Guide is one of a set of Oracle documents that describe how to install, establish, maintain, and use the Oracle7 Server and related Oracle products on OpenVMS.

The README files in the Oracle product directories contain the latest information about the OpenVMS production releases of Oracle7 and related products for OpenVMS.

The purpose of this guide is to help you administer the Oracle7 Server and its application development tools.

For the Latest Information

For the latest information about OpenVMS production releases of Oracle7 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.
This guide is written for system administrators or database administrators (DBAs) who administer Oracle7. This guide assumes that the user has a fundamental knowledge of the OpenVMS operating system; it does not document any features of OpenVMS except when they affect or are affected by the Oracle7 system.

How this Guide Is Organized

This guide is organized as follows:

PART 1: Administering the Oracle7 Server
This part covers activities that are related to maintaining the database once the Oracle7 Server has been installed and the instances accessing the database have been set up.

Chapter 1: Introduction to Oracle7 on OpenVMS
This chapter introduces Oracle products and gives an overview of Oracle7 system on OpenVMS.

Chapter 2: Setting Up Oracle7 Users
This chapter describes how to grant access to Oracle7 and how to set up specific tools.

Chapter 3: Starting Up and Shutting Down Oracle7
This chapter describes how to start up and shut down Oracle7.

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 Oracle7
This chapter describes how to tune Oracle7 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 OpenVMS environment.

Chapter 11: SQL*Plus
This chapter explains how to use SQL*Plus in the OpenVMS environment.

PART III: Appendices
This part includes supplementary information.

Appendix A: National Language Support
This appendix lists the languages, character sets, and sort sequences supported on OpenVMS and describes how to select them.

Appendix B: Logical Names and Parameters
This appendix provides information about Oracle7 logical names and OpenVMS–specific INIT.ORA parameters that are defined differently for different operating systems.

Appendix C: Messages and Codes
This appendix explains the means of error messages you might see while using Oracle products on OpenVMS.

Appendix D: OpenVMS Process Control
This appendix explains how to control Oracle Server processes on OpenVMS.

Appendix E: OpenVMS Supplement to the Generic Documentation Set
This appendix gives OpenVMS specific information for each cross-reference from the generic documentation set.
Conventions Used in this Guide

This section explains the following:

• Syntax
• Terminology

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 Oracle7 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.

Terminology

Oracle Refers only to the company, Oracle Corporation, that develops and distributes the Oracle7 Server and Oracle products.

Oracle7 Server Refers to a single Oracle product. The Oracle7 Server is sometimes referred to as the kernel, the RDBMS, or simply Oracle7. The Oracle7 Server is the Oracle product that stores, manages, and manipulates data. It consists of the database manager, the database administrator utility called Server Manager, several user utilities (Export, Import, SQL*Loader, and the Database Verification Utility), and the data access optimizer.

Oracle7 System Refers to all the Oracle products installed at a site, including the Oracle7 Server and any optional tools such as Oracle Forms or SQL*Net, and to the structures created in shared OpenVMS memory.
when Oracle7 is running. So, the correct definition of an Oracle7 system varies from site to site. Note that in this guide the word database refers only to the files that contain data stored by the Oracle7 Server.

Related Products and Documents

This section explains the following:

- Related products
- Related documents

Related Products

The following is a list of Oracle products, utilities, and options that are available with Oracle7:

- Advanced networking option
- Advanced replication option
- Context
- DBJAV A
- Distributed database option
- National Language Support (NLS) or Multilingual option
- Oracle Intelligent Agent
- Parallel query option
- Parallel server option
- Programmatic Interfaces
- Server Manager
- Spatial Data Option
- SQL*Loader
- SQL*Net
- SQL*Plus
- UTIL

Do Not Link Oracle7 Version 7.1 Products with Oracle7 Server Version 7.3

Products that are distributed with Oracle7 Server Version 7.3.4 or which do not have instructions directing their installation in a 7.3.4 ORA_ROOT must not be installed in a 7.3.4 ORA_ROOT or linked with 7.3.4 libraries. Such products may be used with a 7.3.4 database from
another ORA_ROOT of the appropriate version, using a SQL*Net connection.

**Related Documents**

This documentation set includes the following guides:

- *Oracle7 Server and Tools Administrator’s Guide for OpenVMS* is this guide.
- The *Oracle7 for OpenVMS Installation Guide* for your platform describes installing, upgrading, or migrating Oracle7.
- *SQL*Net Version 2.3.4 for OpenVMS Configuration and User’s Guide describes how to configure and use Network Manager, Protocol Adapters and OpenVMS Clusters.
- *Release Notes* (varies by release)
- README files

For more information on the Oracle7 Server, refer to the following generic documentation:

- The *Oracle7 Application Developer’s Guide* contains specific information required to develop applications for Oracle7.
- *Oracle7 Server Concepts* contains generic information about the Oracle7 Server and describes features and maintenance options available.
- *Oracle7 Server Distributed Systems, Volume I: Distributed Data* describes how to use the distributed data option.
- *Oracle7 Server Distributed Systems, Volume II: Replicated Data* describes how to use the advanced replication option.
- *Oracle7 Server Messages* lists all of the messages and codes that Oracle can return.
- The *Oracle7 Server Reference* describes the Oracle data dictionary tables, initialization parameters, national language support features, and so on.
- The *Oracle7 Server SQL Reference* contains generic information about the Oracle7 SQL language.
- *Oracle7 Server Tuning* shows you how to diagnose performance problems and take corrective action.
• The *Oracle7 Server Utilities* describes the auxiliary utilities provided with the Oracle Server, such as SQL*Loader, the Import utility, and the Export utility.

• *Oracle7 Parallel Server Concepts and Administration* describes the special features of Oracle7 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.
Customer Support Information

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

For Oracle Worldwide Customer Support Services (WCSS), 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 number (CSI) if applicable, or full contact details, including any special project information.
- The complete version numbers of the Oracle7 Server and associated products (for example, Oracle7 Server version 7.3.4 or Oracle Forms version 4.5.6.3.2). Note: The version numbers for products can be obtained from the file ORA_UTIL:PRODUCTS.TXT.
- The hardware type on which the problem occurs (for example, Compaq Alpha).
- The operating system name and version number (for example, OpenVMS 7.1).
- Details of error codes and associated descriptions. Please write these down as they occur, since they are critical in helping WCSS 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 WCSS 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 WCSS 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 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, telephone number, or FAX number.

Compaq Products Division
Publications Manager
Oracle Corporation
500 Oracle Parkway
Box 659104
Redwood Shores, CA 94065
Phone: (650) 506–7000
FAX: (650) 506–7361
# Contents

## PART I  ADMINISTERING THE ORACLE7 SERVER

### Chapter 1  Introduction to Oracle7 on OpenVMS
- Overview of Oracle Products on OpenVMS .................. 1 – 2
- Oracle7 on OpenVMS ........................................... 1 – 3
  - Oracle7 Code ................................................. 1 – 4
  - Oracle7 Instances ........................................... 1 – 4
- Datafiles: Locations and Identifying by Logical Names ... 1 – 8
- Oracle7 Parallel Server Option on OpenVMS ............ 1 – 8

### Chapter 2  Setting Up Oracle7 Users
- Granting Access to Oracle7 Users .......................... 2 – 2
- Using the Oracle7 Password Utility ........................ 2 – 3
- Ending a User’s Session ...................................... 2 – 5

### Chapter 3  Starting Up and Shutting Down Oracle7
- Starting Up the Oracle7 Server ............................. 3 – 1
  - Before Start Up ............................................. 3 – 2
  - Starting Oracle7 via ORACLEINS ......................... 3 – 3
  - Starting Oracle7 via STARTUP Files ..................... 3 – 4
  - Starting Oracle7 via Server Manager .................... 3 – 4
  - Starting Oracle7 Remotely via Server Manager from an OpenVMS Client .................................. 3 – 6
  - Starting Oracle7 Remotely via Server Manager from a Windows PC Client ........................... 3 – 8
Starting Oracle7 Automatically ........................................... 3 – 10
Shutting Down the Oracle7 Server ................................. 3 – 13
Stopping Oracle Users Before Database Shutdown .......... 3 – 13
Shutting Down Oracle7 via ORACLEINS ......................... 3 – 14
Shutting Down Oracle7 via the SHUTDOWN File .............. 3 – 15
Shutting Down Oracle7 via Server Manager ................. 3 – 15
De–installing Shareable Images ..................................... 3 – 16

Chapter 4  Managing Instances .............................................. 4 – 1
Managing Single Instances ............................................. 4 – 2
Components of Instances ................................................. 4 – 2
Controlling Instances ...................................................... 4 – 2
Deleting Instances .......................................................... 4 – 2
Managing Parallel Instances ............................................ 4 – 3
Starting Parallel Instances ................................................ 4 – 3
Issuing Instance Management Commands ..................... 4 – 4
Adding and Enabling Instance Threads ............................. 4 – 7
Preparing an Instance for Parallel Query ....................... 4 – 8
Using the Multi–threaded Server ....................................... 4 – 8

Chapter 5  Managing the Database ........................................... 5 – 1
Introducing Oracle Server Manager ................................. 5 – 2
Server Manager and SQL*Net ........................................... 5 – 2
Starting Server Manager ................................................. 5 – 3
Additions to INIT.ORA When Using Server Manager ........ 5 – 5
SET ECHO and SET TERMOUT Functionality
in SVRMGR ................................................................. 5 – 5
Creating Database Links ................................................... 5 – 6
Creating Multiple Control Files ....................................... 5 – 6
Managing Database Files ............................................... 5 – 7
Using Commands to Manage Database Files .................... 5 – 7
Adding Files ................................................................. 5 – 8
Renaming Files .............................................................. 5 – 8
Moving Tablespace Files ................................................... 5 – 8
Moving Redo Log Files ....................................................... 5 – 10
Database Verification Utility and Other Useful Utilities ........ 5 – 11
Database Verification Utility ............................................. 5 – 11
Other Useful Utilities ....................................................... 5 – 11
Debugging Database Processes with ORAMBX ................ 5 – 13

Chapter 6  Backing Up and Archiving Your Database ............... 6 – 1
Archiving Redo Log Files .................................................. 6 – 2
Chapter 10

Compiling ................................................................. 10 – 9
  Compiler Options Used to Compile Oracle7 ................. 10 – 9
  Floating Point Formats ..................................... 10 – 9
  Pro*Ada, Pro*PL/I, or Pro*Pascal ................................. 10 – 9
  Pro*COBOL .................................................. 10 – 10
Linking ................................................................. 10 – 10
  Syntax .......................................................... 10 – 10
  Example ...................................................... 10 – 11
  Including Option Files .................................... 10 – 11
  Guidelines .................................................. 10 – 12
Using the Oracle Call Interface (OCI) Routines .......... 10 – 15
  Guidelines .................................................. 10 – 16
  CDA/LDA Structure Information .............................. 10 – 16
  Linking OCI Programs Written in C ....................... 10 – 17
  Linking OCI Programs Written in Other Languages .... 10 – 18
  Unexpected Link Errors .................................... 10 – 18
Data Areas and Datatypes ........................................ 10 – 19
Using Literals as Call Arguments .............................. 10 – 19
Optional or Missing Parameters ................................. 10 – 19
Notes on Using Pro*Ada ........................................ 10 – 20
  Post Pro*Ada Install ....................................... 10 – 20
  Relinking Pro*Ada .......................................... 10 – 21
SQL*Module .................................................. 10 – 21
Using Event Flags ................................................ 10 – 22

Chapter 11

SQL*Plus ............................................................... 11 – 1
  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 – 4
    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
Index
Administering the Oracle7 Server
CHAPTER 1

Introduction to Oracle7 on OpenVMS

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

- Overview of Oracle products on OpenVMS
- Oracle7 Server on OpenVMS
Overview of Oracle Products on OpenVMS

The following changes apply to Oracle7.3.2 and later versions on OpenVMS:

- Single-task is no longer supported
- Bequeath adapter is the default adapter
- SQL*Net Version 1 is no longer supported
- SQL*Net Version 2 is the standard connection method
Figure 1 – 1 shows an overview of the Oracle7 Server architecture.

Legend:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCKn</td>
<td>Lock process</td>
</tr>
<tr>
<td>RECO</td>
<td>Recoverer process</td>
</tr>
<tr>
<td>PMON</td>
<td>Process monitor</td>
</tr>
<tr>
<td>SMON</td>
<td>System monitor</td>
</tr>
<tr>
<td>CKPT</td>
<td>Checkpoint</td>
</tr>
<tr>
<td>ARCH</td>
<td>Archiver</td>
</tr>
<tr>
<td>DBWR</td>
<td>Database writer</td>
</tr>
<tr>
<td>LGWR</td>
<td>Log writer</td>
</tr>
</tbody>
</table>

Figure 1 – 1 Oracle7 Server architecture
Oracle7 Code

Oracle7 code consists of several object libraries that are used to form the shareable Oracle7 image during installation; you must link this shareable image with the DEC runtime libraries to produce the executable Oracle7 code.

The code also consists of other shareable images linked during installation. For more information about images, refer to the Oracle7 for OpenVMS Installation Guide for your platform.

Code for the Oracle7 server 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 supports only 32–bit clients. Client applications may not be built with 64–bit pointers.

SVR.EXE File

When the Oracle7 Server is installed as a shareable image, Oracle routines reside as shareable code in OpenVMS global memory.

Products that are distributed with previous versions of the Oracle7 Server should not be linked against the Oracle7 Server Release 7.3.4. However, products already linked with previous versions of Oracle7 should continue to work, connected to an Oracle 7.3.4 database via SQL*Net.

Oracle7 Instances

An Oracle7 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.

Oracle7 Version 7.3.4 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 Oracle7 for 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 3.5 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 Oracle7 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 Oracle7 Server Administrator’s Guide.

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 Oracle7 Server block size. For example, if the blocks are 2 Kb, then the cluster size should be 4Kb, 8Kb, 12Kb, etc. Keep in mind, though, that cluster sizes are specified in terms of disk blocks (where one block = 512 bytes). Thus, a 4Kb 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.

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
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 Oracle7 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. 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 Oracle7 Server Administrator’s Guide.

### Datafiles: Locations and Identifying by Logical Names

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 Oracle7 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.

### Oracle7 Parallel Server Option on OpenVMS

The Oracle Server can run in either exclusive mode or shared (parallel) mode.

<table>
<thead>
<tr>
<th>Oracle + Option</th>
<th>Exclusive Mode</th>
<th>Shared Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Single Node</td>
</tr>
<tr>
<td>OPS not installed</td>
<td>Yes: default</td>
<td>No</td>
</tr>
<tr>
<td>OPS installed</td>
<td>Yes: default</td>
<td>Yes: Single Shared</td>
</tr>
</tbody>
</table>

In *exclusive mode*, only one Oracle instance can mount or open the database. Exclusive mode is necessary to create and completely recover a database. Oracle Server can run in exclusive mode with or without the Parallel Server Option. It is useful to implement OPS in exclusive mode if standard Oracle functionality can meet your current needs, but you want your system to be parallel–server ready.
In *shared mode*, one or more instances of a parallel server mount the same database. All instances mount the database in shared mode and read from and write to the same datafiles. *Single shared mode* describes an Oracle Parallel Server configuration in which only one instance is running. Global operations exist, but are not needed at the moment. The instance operates as though it is in a cluster (with Distributed Lock Manager overhead, and so on), although there is no contention for resources. *Multiple shared mode* describes an Oracle Parallel Server configuration with multiple instances running.

**Note:** “Shared” mode is also known as “parallel” mode. There is no difference between the options PARALLEL and SHARED in either the ALTER DATABASE statement or the STARTUP command.

Figure 1 – 2 illustrates a typical configuration of Oracle running in shared mode with three instances on separate nodes accessing the database.

![Figure 1 – 2 Shared Mode Sharing Disks](image-url)
Oracle Parallel Server Architecture

The Oracle7 Parallel Server technology runs on an OpenVMS Cluster configuration and allows multiple Oracle7 instances to access a single logical and physical database on a shared disk system.

When you directly access a database with a single instance, you are running in exclusive mode. When you directly access a database with two or more instances, you are running in parallel mode. The Parallel Server technology enables you to run in parallel mode.

**Note:** A database created on a VAX cannot be opened on an Alpha and vice versa. Furthermore, a parallel server configuration can consist of any number of instances so long as they are on the same type of node (either all Alpha or all VAX).

OpenVMS Cluster Technology

OpenVMS Cluster technology is Compaq’s extension to its hardware line and OpenVMS operating system. An OpenVMS Cluster configuration enables multiple Compaq computers or nodes to share access to data, software, and peripheral devices. Each node on an OpenVMS Cluster configuration operates independently of other nodes, except when sharing a resource such as a data file or a print queue. The OpenVMS Distributed Lock Manager is used to coordinate access to shared resources between nodes.

The challenge in implementing a DBMS to fully utilize an OpenVMS Cluster configuration is the use of buffer caches by the DBMS. Each node on an OpenVMS Cluster configuration has its own memory, which is not shared with any other node. The difficulty with implementing a DBMS that fully uses OpenVMS Cluster technology is efficiently coordinating buffer caches between nodes on an OpenVMS Cluster without sacrificing functionality or performance. The Oracle7 Parallel Server technology achieves efficient buffer cache coordination using parallel cache management.

Parallel Cache Management

Each Compaq system can run one or more instances of the Oracle7 Parallel Server technology. Each instance serves multiple clients, has its own private memory for database buffers, and has its own set of backup and recovery processes. Using parallel cache management, the Oracle7 Parallel Server technology gives multiple instances direct, shared access to the same database on disk.

The Oracle7 Parallel Server technology coordinates multiple buffer caches by using the OpenVMS Distributed Lock Manager to keep track of the state and location of the data in each cache. If one instance needs to modify data that has been modified in another instance’s buffer cache, the Lock Manager causes the second instance to write the data to disk so that the first instance can access it.
The Oracle7 Parallel Server technology uses its own internal mechanisms for concurrency control (for example, to lock database objects such as rows and tables). The OpenVMS Distributed Lock Manager is used only for communication and cache coordination between instances. The Oracle Parallel Server technology reads only data blocks from disk that are not already in the buffer cache of the instance that needs the data. Modified data blocks are written to disk only when they are needed by another instance, when the buffer cache space they occupy is needed by more recently requested data, or for periodic checkpoints. When a transaction commits, at most one I/O to the redo log file is needed to ensure data integrity. Thus, fast commits, group commits, and deferred writes are fully supported in the Oracle7 Parallel Server technology.

When using the Oracle7 Parallel Server technology, all OpenVMS systems running an instance do database work. If one node on the OpenVMS Cluster system becomes unavailable, affected users can switch to other nodes and continue processing. An instance on a remaining node automatically performs recovery for the failed node. This minimizes the amount of time that users are denied access to their data by system failures.

Compaq provides many ways for member nodes in OpenVMS clusters to communicate. These cluster interconnects span a wide range of speed and throughput. When using Oracle7 Parallel Server technology in an OpenVMS cluster, you should make sure that the cluster interconnect between nodes running Parallel Server is as fast as possible for the hardware types of the nodes in question. The higher the speed of the interconnect and the greater the throughput, the better Oracle Server will perform. Note that one node on a slow interconnect could act as a bottleneck for the other nodes participating in a Parallel Server environment.
Setting Up Oracle7 Users

This chapter assumes that you have installed Oracle7 and have created a database and started an instance. This chapter describes how to set up Oracle7 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 Oracle7 users
- Using the Oracle7 Password Utility
- Ending a user’s session
Granting Access to Oracle7 Users

When the Oracle7 Server is installed with the Group option, all Oracle7 users must belong to the same group that includes the Oracle7 account. Otherwise, the Oracle7 Server is available to the entire system.

To grant users access to Oracle7, 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 Oracle7 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.V7.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.V7.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 *Oracle7 for OpenVMS Installation Guide* for your platform.

3. Create the Oracle7 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 *Oracle7 Server Administrator’s Guide*.

4. If you have a user who uses Server Manager to start up or shut down an Oracle7 instance, use the OpenVMS AUTHORIZE utility 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 the *Oracle7 for OpenVMS Installation Guide* for your platform.

---

### Using the Oracle7 Password Utility

On OpenVMS, you do not create a password file as part of the Oracle7 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 Oracle7 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 Oracle7 Server 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 neither DECnet nor TCP/IP are truly secure protocols, Oracle Corporation strongly recommends that you make use of the password utility for these protocols.

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 -
DISK:[DIRECTORY]FILENAME.TYPE
```

**Note:** During system startup, this logical name needs to be redefined prior to starting Oracle.

For example:

```
$ DEFINE/SYSTEM/EXEC ORA_PAYROL_PWFILE -
DISKS$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 Oracle7 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/PARALLEL).

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 Oracle7 Server Administrator’s Guide.

Ending a User’s Session

You can end a user’s Oracle7 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.

Note: You should only use one of the above methods to end a specific Oracle7 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 Oracle7

This chapter describes different ways to start up or shut down the Oracle7 Server. These methods include using ORACLEINS, using STARTUP and SHUTDOWN files, or using Server Manager.

The following topics are presented:

- Starting up the Oracle7 Server
- Shutting down the Oracle7 Server
Starting Up the Oracle7 Server

Before you can start the Oracle7 Server, both an instance and a database must exist on your local system. If you did not install the Oracle7 Server, consult the person who did.

This section presents the following topics:

- Before start up
- Starting Oracle7 via ORACLEINS
- Starting Oracle7 via STARTUP files
- Starting Oracle7 via Server Manager
- Starting Oracle7 remotely via Server Manager from an OpenVMS client
- Starting Oracle7 remotely via Server Manager from a Windows PC client
- Starting Oracle7 automatically

**Before Start Up**

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

After rebooting OpenVMS, you must perform the following steps before starting the Oracle7 Server:

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

3. If you plan to install shared images, run the file ORA_UTIL:INSUTILITY.COM. (You might have to create this file by running ORA_INSTALL:ORA_INSUTL.COM.)

INSUTILITY.COM installs Oracle products other than the Oracle7 Server (such as Oracle Forms and SQL*Plus) in shared memory.

You must perform steps 2 and 3 under the following conditions:

- After the ORACLE image has been removed, but before the Oracle7 Server 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 Oracle7 via ORACLEINS

To start Oracle7 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:
Starting Oracle7 via
STARTUP Files

You can also use command files to start Oracle7. 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 one of the following STARTUP command files for the instance you want to start:

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

or:

$ @ORA_DB:STARTUP_PARALLEL_<dbname>.COM <sid> <setup_nodename>

These files are 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 Oracle7 via
Server Manager

You can also start an instance of Oracle7 using Server Manager. See page 5–2 of this manual for instructions on setting up Server Manager on your 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 Oracle7 Server, you start up the current instance. The current Oracle7 instance is identified by the value of the logical name ORA_SID. For example, if the value of ORA_SID is currently V7343, the current instance is the instance with the SID V7343. 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 Oracle7 Server 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 Oracle7 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. You must include the directory location of the file:

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

To start Oracle7, you must have the process rights identifier ORA_DBA or ORA_<sid>_DBA assigned to your user account in the OpenVMS rights database and you must run the .COM file that makes the logical name assignments required to run Oracle7.

Before starting up Oracle7, 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 Oracle7 Server Administrator’s Guide. You can issue the single Server Manager command, STARTUP, or execute the three separate Server Manager commands documented in the Oracle7 Server Administrator’s Guide to start the Oracle7 Server.

The Server Manager command STARTUP starts the current ORACLE instance, creating the SGA in OpenVMS shared memory and creating the detached processes. It then mounts the database and opens it.
Starting Oracle7 Remotely via Server Manager from an OpenVMS Client

You can use Server Manager on an OpenVMS client to start up an Oracle7 database instance on a remote 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. 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. Edit the `ORASRV_NETV2.COM` pointed to in the `LISTENER.ORA` file for this `<sid>`. Add logical name definitions for `ORA_DB`, `ORA_CONTROL1`, `ORA_CONTROL2`, `ORA_PARAMS`, `ORA_INITSQL`, and `ORA_ARCHIVE` just above the line that invokes `ORASRV`. These definitions should look similar to the following (you can cut and paste these definitions from the file `ORA_DB:ORA_DB_<database name>.COM`):
   
   ```
   $ define/nolog/job ORA_DB –
   ddcn:[<oracle root directory>.db_<database name>]
   ```
   
   ```
   $ define/nolog/job ORA_CONTROL1 –
   ```
ORA_DB:ORA_CONTROL1.CON

$ define/nolog/job ORA_CONTROL2 -
ORA_DB:ORA_CONTROL2.CON

$ define/nolog/job ORA_PARAMS -
ORA_DB:<nodename>_<sid>_INIT.ORA

$ define/nolog/job ORA_INITSQL ORA_SYSTEM:SQL.BSQ

$ define/nolog/job ORA_ARCHIVE -
ddcn:[<oracle root directory>.db_<database name>]

6. **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 from the SID on the remote system where the database resides.**

2. **Define the process logical name ORA_DFLT_HOSTSTR to the SQL*Net V2 alias for the remote system.** For example:

   $ DEFINE ORA_DFLT_HOSTSTR <SQL*Net V2 alias>

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_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 2.3.4.0.0 - Production

   Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.
Starting Oracle7 Remotely via Server Manager from a Windows PC Client

Oracle7 Server Release 7.3.4.3.0 – Production
PL/SQL Release 2.3.4.3.0 – Production
SVRMGR> connect internal as sysdba
Password:
Connected to an idle instance.
SVRMGR> startup
ORACLE instance started.
Total System Global Area  8071392 bytes
Fixed Size               53216 bytes
Variable Size           6859008 bytes
Database Buffers        1126400 bytes
Redo Buffers            32768 bytes
Database mounted.
Database opened
SVRMGR>

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

You can use Server Manager on a Microsoft Windows PC to start up an Oracle7 database instance on a remote 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. 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 -
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. Edit the ORASRV_NETV2.COM pointed to in the LISTENER.ORA file for this <sid>. Add logical name definitions for ORA_DB, ORA_CONTROL1, ORA_CONTROL2, ORA_PARAMS, ORA_INITSQL, and ORA_ARCHIVE just above the line that invokes ORASRV. These definitions should look similar to the following:

   $ define/nolog/job ORA_DB -
   ddcn:[<oracle root directory>.db_<database name>]

   $ define/nolog/job ORA_CONTROL1 ORA_DB:ORA_CONTROL1.CON

   $ define/nolog/job ORA_CONTROL2 ORA_DB:ORA_CONTROL2.CON

   $ define/nolog/job ORA_PARAMS -
   
   ORA_DB: <nodename>_<sid>_INIT.ORA

   $ define/nolog/job ORA_INITSQL ORA_SYSTEM:SQL.BSQ

   $ define/nolog/job ORA_ARCHIVE -
   ddcn:[<oracle root directory>.db_<database name>]

6. Copy ORA_DB:<nodename>_<sid>_INIT.ORA, 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 SQLNET V2 alias entry for the SID on the remote system where the database resides.

2. Edit the <nodename>_<sid>_INIT.ORA, INIT.ORA, and INITPS.ORA files that were copied from the server and change all the IFILE parameters to point to the local WINDOWS path to which these files were copied.
3. Invoke SVRMGR from Windows Explorer (Windows 95/98/NT). It should be located in the following directory: \ORAWIN95\BIN for Windows95, \ORAWIN98\BIN for Windows98, 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_V2_ALIAS is the TNSNAMES.ORA alias for the remote database.

Oracle Server Manager Release 2.3.4.0.0 – Production

Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.

Oracle7 Server Release 7.3.4.3.0 – Production Release
PL/SQL Release 2.3.4.3.0 – Production

SVRMGR> connect internal@<sqlnet_V2_alias>
Password:
Connected to an idle instance.
SVRMGR> startup pfile=<WINDOWS path to <node>_<sid>_INIT.ORA>
ORACLE instance started.
Database mounted.
Database opened.
Total System Global Area 8071392 bytes
  Fixed Size 53216 bytes
  Variable Size 6859008 bytes
  Database Buffers 1126400 bytes
  Redo Buffers 32768 bytes
Database mounted.
Database opened.
SVRMGR> exit
Server Manager complete.

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

Starting Oracle7 Automatically

To start Oracle7 automatically whenever you start OpenVMS, submit the Oracle7 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 Oracle7
• Run as the operating system DBA account user
• Run ORA_RDBMS:INSORACLE.COM to install the global sections required by Oracle7
• Run ORA_UTIL:INSUTILITY.COM if you previously ran ORA_INSTALL:ORA_INSUTIL.COM to install Oracle products as shared images
• Execute one of the startup command files to start Oracle7:
  $ @ORA_DB:STARTUP_EXCLUSIVE_<dbname>.COM
  or
  $ @ORA_DB:STARTUP_PARALLEL_<dbname>.COM

Sample Startup File
A sample startup file that starts two Oracle7 systems automatically after a system reboot is shown below:

$! STARTORAV7.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 version of ORACLE
$! code by running the appropriate ORAUSER.COM:
$!
$! $ @DISK$ORACLE:[ORACLE.V7.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 Oracle7 code. The example assumes that the Oracle7 root directory is DISK$ORACLE: [ORACLE.V7].

Run this file as a batch job under the Oracle7 account as part of the standard system startup procedure. Keep this file in the Oracle7 account login directory.

For example, if the Oracle7 account resides in DISK$ORACLE: [ORACLE], and the startup script is named STARTORAV7.COM, then you would start this script at boot time by adding the following lines to SYS$MANAGER: SYSTARTUP_VMS.COM:

```
filspc = "DISK$ORACLE: [ORACLE]STARTORAV7"
submit -
/user=Oracle7 -
/after="+00:05:00" -
/log='filspc'.log -
'filspc'
```
Shutting Down the Oracle7 Server

To shut down Oracle7, you can use one of these methods:

- Stopping Oracle Users Before Database Shutdown
- Shutting down Oracle7 via ORACLEINS
- Shutting down Oracle7 via the SHUTDOWN file
- Shutting down Oracle7 via Server Manager
- De–installing shareable images

After all instances on a node have been shut down, you must de–install the shareable images.

This section describes the three methods of shutting down Oracle7 and then tells how to deinstall shareable images.

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 still appears to 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;

<table>
<thead>
<tr>
<th>SID</th>
<th>SERIAL#</th>
<th>PROCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>20C0018B</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>20C0018C</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>20C0018D</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>20C0018E</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>20C0018F</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>20C002DD</td>
</tr>
</tbody>
</table>

6 rows selected.

Process 2 with OS–process id 20C00470
$ sqlplus <un>/<pw>
Process 1:

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

<table>
<thead>
<tr>
<th>SID</th>
<th>SERIAL#</th>
<th>PROCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>20C0018B</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>20C0018C</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>20C0018D</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>20C0018E</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>20C0018F</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>20C002DD</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>20C00470</td>
</tr>
</tbody>
</table>

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 Oracle7 via ORACLEINS**

To shut down Oracle7 using ORACLEINS:

1. Using Server Manager, ensure that there are no open sessions.
2. Run the database–specific ORAUSER file:
   ```
   $ @ORA_DB:ORAUSER_<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 Oracle7 via the SHUTDOWN File**

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

```sh
$ @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 Oracle7 systems automatically is shown below:

```sh
$!
$! NAME: STOPORAV7.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.
$!------------------------------------------------------------
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
$!
```

Starting Up and Shutting Down Oracle7 3 – 15
You can shut down an instance of Oracle7 using Server Manager. See page 5–2 in this manual for instructions on setting up Server Manager on your OpenVMS platform. Then, refer to the generic (platform–independent) Oracle Server documentation for instructions on using Server Manager.

After shutting down all Oracle7 instances on a node, perform the following steps:

1. Before relinking, deinstall the shareable images by entering the following statement:
   $$ @ORA_RDBMS:REMORACLE.COM $$

2. If you ran ORA_INSTALL:ORA_INSUTL.COM or ORA_UTIL:INSUTILITY.COM to install Oracle products into shared memory, remove the products by entering the following command:
For more information about these utilities, see Chapter 8 of this Guide.
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
- Managing parallel instances
- Preparing an instance for Parallel Query
- Using the multithreaded server
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.
5. Clean out the instance’s trace files in ORA_DUMP.
Managing Parallel Instances

This section discusses the following topics:

- Starting parallel instances
- Issuing instance management commands
- Adding and enabling instance threads

Refer to the *Oracle7 Parallel Server Concepts and Administration* for more information on instance management.

Starting Parallel Instances

To start instances in parallel mode, follow this procedure:

1. Shut down the initial instance, if applicable.

   **Warning:** 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.

2. On every node that will have an instance running, execute the following command, if it has not already been run:

   `$ @ORA_RDBMS:INSORACLE.COM`

3. Run the appropriate startup command:

   3.1 If you set up each instance on the node *where it will eventually run*, execute the command:

      `$ @ORA_DB:STARTUP_PARALLEL_<dbname>`

      **Note:** Oracle Corporation recommends that you always qualify this script with the instance’s SID.

   3.2 If you set up more than one instance on the node, qualify the command with the instance’s SID, and execute the command from the node *where the instance was set up*:

      `$ @ORA_DB:STARTUP_PARALLEL_<dbname> <sid>`

   3.3 If you set up additional instances on the same node but want to run an instance on a different node, qualify the command with the instance’s SID and the node where the instance was set up.

      Execute the following command from the node *where you want the instance to run*:

      `$ @ORA_DB:STARTUP_PARALLEL_<dbname> <sid> <setup_nodename>`

At this point, you have set up your system to run in parallel mode.
Issuing Instance Management Commands

This section describes the relationship between the node where you set up an instance (using ORACLEINS) during the parallel mode setup procedures, and the node where you would issue that instance’s management command files (STARTUP, SHUTDOWN, or ORAUSER). You can issue these management commands on different nodes, depending on which node you want the instance to reside on.

- If instance A was set up on node1 and instance A is the only instance on node1, then you execute instance A’s command without a qualifier on node1.
- If instance A and instance B were both set up on node1 with instance A set up first and instance B set up second, then you execute instance A’s command qualified with A’s SID on node1, and you execute instance B’s command qualified with B’s SID on node1.
- If instance C was set up on node2 and instance C is the only instance on node2, then you execute instance C’s command without a qualifier on node2.
- If instance A, instance B, and instance C were set up on node1 but you want instance C to reside on node2, then you execute instance A’s command qualified with A’s SID on node1, execute instance B’s command qualified with B’s SID on node1, and execute instance C’s command qualified with C’s SID and setup node (node1) on node2.

Example 1

When executing ORACLEINS during the parallel mode setup, you set up three instances to access a database named MIS as shown in this table:

<table>
<thead>
<tr>
<th>Database</th>
<th>Node</th>
<th>Instance Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>HARPO</td>
<td>MKT1, MKT2</td>
</tr>
<tr>
<td>MIS</td>
<td>ZEPPO</td>
<td>MKT3</td>
</tr>
</tbody>
</table>

You want to start up the instances on the node on which they were set up, in parallel mode. You also want to define the logical names and symbols for each instance, using ORAUSER. Proceed as follows:
1. Execute instance MKT1’s commands qualified with MKT1’s SID from HARPO.

<table>
<thead>
<tr>
<th>Database</th>
<th>Node</th>
<th>Instance Setup</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>HARPO</td>
<td>MKT1, MKT2</td>
<td>$ @ORAUSER_MIS MKT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$ @STARTUP_PARALLEL_MIS MKT1</td>
</tr>
</tbody>
</table>

Because there is more than one instance on HARPO, you should always qualify the commands with the SID.

2. Execute instance MKT2’s commands qualified with MKT2’s SID from HARPO.

<table>
<thead>
<tr>
<th>Database</th>
<th>Node</th>
<th>Instance Setup</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>HARPO</td>
<td>MKT1, MKT2</td>
<td>$ @ORAUSER_MIS MKT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$ @STARTUP_PARALLEL_MIS MKT2</td>
</tr>
</tbody>
</table>

Because MKT2 was set up second on HARPO, qualify the commands.

3. Execute instance MKT3’s commands without a qualifier from ZEPPO.

<table>
<thead>
<tr>
<th>Database</th>
<th>Node</th>
<th>Instance Setup</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>ZEPPO</td>
<td>MKT3</td>
<td>$ @ORAUSER_MIS MKT3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$ @STARTUP_PARALLEL_MIS</td>
</tr>
</tbody>
</table>

To start up and define logical names for MKT3 on ZEPPO, you do not need to qualify the commands; however, Oracle Corporation recommends that you do so as a safety precaution.

Example 2 When executing ORACLEINS during the parallel mode setup, you set up three instances to access a database named MIS, as shown in this table:

<table>
<thead>
<tr>
<th>Database</th>
<th>Node</th>
<th>Instance Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS</td>
<td>HARPO</td>
<td>MKT1, MKT2, MKT3</td>
</tr>
</tbody>
</table>

You want to start up MKT1 and MKT2 on HARPO and start up MKT3 on ZEPPO. All three instances are started in parallel mode. You also want to define the logical names for each instance. Proceed as follows:
1. Execute instance MKT1’s commands qualified with MKT1’s SID from HARPO.

<table>
<thead>
<tr>
<th>Node</th>
<th>Instance Setup</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARPO</td>
<td>MKT1, MKT2, MKT3</td>
<td>$ @ORAUSER_MIS MKT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$ @STARTUP_PARALLEL_MIS MKT1</td>
</tr>
</tbody>
</table>

   Because there is more than one instance on HARPO, you should always qualify the commands with the SID. Execute the commands from HARPO.

2. Execute instance MKT2’s commands qualified with MKT2’s SID from HARPO.

<table>
<thead>
<tr>
<th>Node</th>
<th>Instance Setup</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARPO</td>
<td>MKT1, MKT2, MKT3</td>
<td>$ @ORAUSER_MIS MKT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$ @STARTUP_PARALLEL_MIS MKT2</td>
</tr>
</tbody>
</table>

   Because MKT2 was set up second on HARPO, you have to qualify the commands. Execute the commands from HARPO.

3. Execute instance MKT3’s commands qualified with MKT3’s SID and setup node from ZEPPO.

<table>
<thead>
<tr>
<th>Node</th>
<th>Instance Setup</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARPO</td>
<td>MKT1, MKT2, MKT3</td>
<td>$ @ORAUSER_MIS MKT3 HARPO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$ @STARTUP_PARALLEL_MIS MKT3 HARPO</td>
</tr>
</tbody>
</table>

   When starting up an instance on a node that is not the setup node, you should always make sure that the instance is not already started on the setup node.

   **Note:** STARTUP_EXCLUSIVE_MIS.COM and SHUTDOWN_MIS.COM operate in the same manner. Examples for these scripts would look almost identical to those given above.
Adding and Enabling Instance Threads

The definition of a symbol that sets up your ZEPPO users to use instance MKT3 could look like one of the following, depending on where you set up the instance. The first reflects Example 1 and the second reflects Example 2.

\[
\text{MKTMIS3} \equiv \text{"@DISK$MIS:[ORACLE.V7.DB_MIS]ORAUSER_MIS MKT3"}
\]

or

\[
\text{MKTMIS3} \equiv \text{"@DISK$MIS:[ORACLE.V7.DB_MIS]ORAUSER_MIS MKT3 HARPO"}
\]

With Oracle7, 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:

\[
\text{ALTER DATABASE ADD LOGFILE THREAD <number> <filespace> <storage>};
\]

For example:

\[
\text{ALTER DATABASE ADD LOGFILE THREAD 2}
\]

\[
\]

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.

\[
\text{ALTER DATABASE ENABLE PUBLIC THREAD <number>}
\]

For example:

\[
\text{ALTER DATABASE ENABLE PUBLIC THREAD 4;}
\]

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 *Oracle7 Server Administrator’s Guide*.

---

### Using the Multi–threaded Server

The Oracle7 Server running in a multi–threaded server configuration might run into the DECnet “known object” limit. The limit is system–dependent. To see the limit on your system, do the following:

1. Start DECnet’s NCP facility
2. Issue the command SHOW EXEC CHAR
3. Look for the value of MAXIMUM DECLARED OBJECTS

From NCP, you can also find how many known objects are currently on your system by using the SHOW KNOWN OBJECTS command. In the output to this command, objects in the format ORA_NTD<xxxx>_1 are dispatcher processes. Objects with names specified in the LISTENER.ORA files are listener processes.
Ensuring that the Oracle7 Server operates successfully can involve tuning the system or modifying parameters. These tasks require a thorough understanding of OpenVMS system administration as well as the concepts documented in the Oracle7 Server Administrator’s Guide.

This chapter presents the following topics:

- Introducing Oracle Server Manager
- Creating database links
- Creating multiple control files
- Managing database files
- Database verification utility and other useful utilities
- Debugging database processes with ORAMBX
Introducing Oracle Server Manager

Oracle Server Manager is Oracle Corporation’s graphical database administration tool. Using Server Manager, you can perform database administration tasks with the ease of a graphical user interface (GUI).

If your environment does not support a GUI, you can use Server Manager in line mode. Line mode allows you to execute commands explicitly on a command line. Server Manager in line mode is especially useful when a graphical device is unavailable, or for unattended operations, such as running batch jobs or scripts which do not require user intervention.

Server Manager is an important tool for observing the performance of the Oracle Server on your system. You can also use it to monitor performance in a distributed environment. You can also use Server Manager to connect to other databases.

Use Server Manager to:

• Start up, shut down, back up, administer, and recover a database
• Perform multiple tasks, administrative and non-administrative, simultaneously, using Server Manager’s multiple windows
• Administer multiple databases simultaneously, including remote databases running on any Oracle platform at any location worldwide
• Dynamically execute, and keep a record of, SQL statements

By default, Server Manager can only be used by DBAs. To enable someone who does not have the ORA_DBA rights identifier to use Server Manager, the system’s administrator must grant permission.

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 Version 2.3.4 for OpenVMS Configuration and User’s Guide.

On OpenVMS systems, Server Manager is linked two-task. This requires Server Manager to make connections to the Oracle7 Server using SQL*Net. The SQL*Net V2 listener process must be running on the server machine. If the SQL*Net listener is not running, you will receive the following errors when trying to connect if you are not using the bequeath adapter:
GUI mode

MGR-02703: an error occurred when connecting to a database
ORA-12546: TNS: permission denied

Line mode

ORA-12546: TNS: permission denied

Starting Server Manager

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

SVRMGR GUI-mode

Issue the OpenVMS command SHOW DISPLAY to make sure that your display is set correctly. If your display is not set correctly, the following error occurs:

$DECW–W–OPENIN, error opening DECW$DISPLAY as input
–SYSTEM–W–NOSUCHDEV, no such device available

If your display is not set before invoking SVRMGR, the following error will occur:

TK2–04097: Oracle Toolkit II’s connection to the window system was refused

To set your display, use the SET DISPLAY command as follows:

• TCP/IP

$ SET DISPLAY/CREATE/TRANSPORT=TCP/IP/NODE=<your IP address>

• DECnet

$ SET DISPLAY/CREATE/NODE=<your DECnet nodename>

When starting Server Manager in GUI mode (that is, running under Motif), implement one of the following steps before invoking SVRMGR:

• Provide a valid SQL*Net V2 “Service Name” in the SVRMGR “Connect” Window

or

• Define the logical name ORA_DFLT_HOSTSTR to a valid SQL*Net V2 service name

or
• Use the bequeath adapter

If you enter an invalid “Service Name”, you will receive the following errors:

MGR-02703: an error occurred when connecting to a database
ORA-12154: TNS: could not resolve service name

To invoke Server Manager on Motif, enter the following command:

$ SVRMGR

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 V2 service name

or

• Use the bequeath adapter

To invoke Server Manager, enter the following command:

$ SVRMGR

Defining Logical Names

The SQL*Net task files used by SVRMGR to connect to and manage the databases need to define more logical names than those used for normal user connections. Since SVRMGR ALWAYS uses SQL*Net to connect to databases, it is critical that you identify these task files and possibly separate them from those used by normal users.

In your SVRMGR SQL*Net task files, you need to define all the logical names in ORA_DB:ORA_DB_<dbase_name>.COM, as follows:

• ORA_DB
• ORA_ARCHIVE
• ORA_CONTROL1
• ORA_CONTROL2
• ORA_PARAMS
• ORA_INITSQL
These logical names need to be defined the same way they are defined in the file ORA_DB:ORA_DB_<dbase_name>.COM, making sure that the definitions use the qualifier /JOB.

The simplest implementation is to run ORA_DB:ORA_DB_<dbase_name>.COM in your SVRMGR SQL*Net task files. This approach increases process startup times and increases the memory requirements of the SQL*Net server processes, which is usually not a problem. However, if it is a problem, you will need to define these logical names in your “lite” version of the SQL*Net task files with the /JOB qualifier for SVRMGR.

In addition to the previously listed logical names, you need to define ORA_COMMON in the SQL*Net task files used by SVRMGR. Otherwise, SVRMGR cannot be used successfully for most database management activities. Make sure that ORA_COMMON points to your RDBMS directory ORA_RDBMS. Be sure to use the full path specification with no process–level or job–level concealed logical names like ORA_ROOT.

Caution: To administer a database over SQL*Net, that is connected as SYSDBA or SYSOPER, you must have done the following:

• Created a password file for your database
• Defined the necessary logical name
• Defined the password REMOTE_LOGIN_PASSWORDFILE in the database file INIT.ORA

Failure to complete any of these steps will prevent successful remote administration of your database.

Additions to INIT.ORA When Using Server Manager

If you are going to use SVRMGR to manage your database, you must add the DB_NAME= parameter to your INIT.ORA file to explicitly identify the database name. While the scripts generated when you created a new database identified the database to SQL*DBA explicitly in the past, currently the SQL*Net connections used by SVRMGR receive no such information.

SET ECHO and SET TERMOUT Functionality in SVRMGR

In 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
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 Version 2 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
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 Oracle7 Server Administrator’s Guide for general information. Specific information for 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 Oracle7 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, back up the control files as in the following example:

SVRMGR> ALTER DATABASE RENAME FILE
2> 'DISK$1:[ORACLE.TEST.DB_V7TEST]ORA_SYSTEM.DBS' TO
3> 'MY$DISK:[ORACLE.TEST.DB_V7TEST]ORA_SYSTEM.DBS'
$ BACKUP/LOG/VERIFY –
DISK$1:[ORACLE.TEST.DB_V7TEST]*.CTL –
MY$DISK:[ORACLE.TEST.DB_V7TEST]*.CTL

Note: The physical device name and the file location must appear exactly as in the control file. If you cannot bring up the database, use DUMP to get the physical device name and the database file location(s) (for example, $ DUMP <control_file>). If you can bring up the database, enter the following commands to get the physical device name and the database file location(s):

$ SVRMGR
SVRMGR> CONNECT INTERNAL
SVRMGR> SELECT * FROM DBA_DATA_FILES;
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.

$ SVRMGR
SVRMGR> CONNECT INTERNAL
SVRMGR> SELECT * from DBA_DATA_FILES;
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.
   $ SVRMGR
   SVRMGR> CONNECT INTERNAL
   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
     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.
   ```sql
   $ SVRMGRL
   SVRMGR> CONNECT INTERNAL
   SVRMGR> SHUTDOWN
   SVRMGR> EXIT
   $ BACKUP/IGNORE=(INTERLOCK,NOBACK) -
   <old_device>:[<dir>]<filename>.<ext> -
   <new_device>:[<new_dir>]<new_filename>.<ext>
   $ SVRMGRL
   SVRMGR> CONNECT INTERNAL
   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.
   ```sql
   SVRMGR> ALTER DATABASE RENAME FILE
   2> '<device>:[<dir>]<old_redofile>.RDO',
   3> '<device>:[<dir>]<old_redofile2>.RDO' to
   4> '<device>:[<dir>]<new_redofile>.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, results will be unpredictable.

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
   
   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. This utility comes with Oracle7 Server Version 7.3 and above. Invoke this utility with the `DBV` symbol on OpenVMS.

To use this utility to verify data in an Oracle7 Version 7.1 database, point to the 7.1 database files from your Oracle7 Server Version 7.3 installation.

**Additional Information:** Refer to the *Oracle7 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>]

TSLDM

Caution: USE WITH EXTREME CARE!

This program can be used to dump information on the locks present on your system, in a manner that may be easier to read than the display provided by SDA. It will display locks held in user (the default), supervisor, executive, or kernel–mode, selected by a variety of criteria.

$ TSLDM <option>

where the options are:

b – Show only blocking locks
n – Sort by parent lock id then lock id
p – Show process name
o=prefix – Show only this prefix (default=show all)
m=pattern – Show only this pattern (wildcards allowed)
s,e – Execute in executive mode
k – Execute in kernel mode
v – Show lock values (132 columns)
f=(a,b,c) – Show all locks for a file id (kernel mode)
r – Print locks in random order (no sort)
t – Print only totals
? – Display the above list

EXAMPLE: Here is some default output from the TSLDM utility when called with no arguments:

$ TSLDM
LOCK DATABASE: USER MODE

Sorting locks...
### 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

---

<table>
<thead>
<tr>
<th>RESOURCE</th>
<th>LOCKID</th>
<th>PARENT</th>
<th>RO/GR/Q</th>
<th>PROCESS</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBG$ALIVE_4BA00451</td>
<td>6900AA47</td>
<td>PW/NL/W</td>
<td></td>
<td>DBG$4BA00451</td>
<td>DONALD</td>
</tr>
<tr>
<td>DBG$ALIVE_4BA00451</td>
<td>2A00AA4F</td>
<td>NL/EX/G</td>
<td></td>
<td>DONALD</td>
<td>DONALD</td>
</tr>
<tr>
<td>JBC$DUCKY</td>
<td>03000246</td>
<td>EX/EX/G</td>
<td></td>
<td>JOB_CONTROL</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>JBC$CHECK_DB</td>
<td>05000242</td>
<td>PR/PR/G</td>
<td></td>
<td>JOB_CONTROL</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>JBC$CHECK_DB</td>
<td>05000242</td>
<td>NL/NL/G</td>
<td></td>
<td>JOB_CONTROL</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>QMAN$MSR_DSA1x00x00x00x00x00x00x00x</td>
<td>05000242</td>
<td>NL/NL/G</td>
<td></td>
<td>JOB_CONTROL</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>RMC$SHADOW_SYNC</td>
<td>0300024D</td>
<td>CR/CR/G</td>
<td></td>
<td>SHADOW_SERVER</td>
<td>SYSTEM</td>
</tr>
</tbody>
</table>

6 locks processed, 6 dumped

END OF LOCK DATABASE

$
BLOCK <dbuser> <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. Running ORAMBX against the client application will result in an error from ORAMBX. This is normal.
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 *Oracle7 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 Oracle7 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 *Oracle7 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 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 *Oracle7 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.V7_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.

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 step 2, 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>
```

This command archives the instance’s current redo log file. 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 and LOG_ARCHIVE_DEST 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 you want to assign to the log.

- 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:

- Back up a closed database (offline backup)
- Back up an open database (online 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 OpenVMS utility BACKUP. The Oracle7 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.

Attention: 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.

If the warning message “File locked by another user” appears during the backup procedure, you can ignore this message and continue the backup.

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 SYS$ERROR, not SYS$OUTPUT and can be saved to file if you use the LOGFILE option.

You can export the entire database or portions of the database. You can also perform incremental exports, which save only tables that changed since the last export; these exports are quicker and more convenient. To restore the export file generated by the Export utility, use the Import utility. For information about using these utilities, refer to the *Oracle7 Server Utilities*.

Note that under 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>
```

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 the 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.
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 the 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 Oracle7 database is the process of restoring normal Oracle7 operations when they are interrupted by an 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 Oracle7 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 an 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 Oracle7 Server Administrator’s Guide.

Instance recovery is done automatically whenever an instance is started. It can be performed after an 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 Oracle7 Server Administrator’s Guide and to the Oracle7 Server Utilities for information about the Oracle7 utilities used in recovery procedures.
Recovering from an 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. An instance failure can be caused by loss of power, machine malfunction, an operating system crash, or another hardware or software problem. You can diagnose an 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 an 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 Oracle7 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 an instance failure. If possible, the problem that caused the 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 a 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 a media failure. If you prepared for a media failure properly, you can restore the following:

- 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 Oracle7 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 perform a media recovery. If you have suffered from a 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 an Export File

Refer to the Oracle7 Server Utilities for information on how to restore from an export file as part of a media recovery. If you decide to import from an export file as part of a 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 Oracle7 as SYSTEM and run the CATDBSYN.SQL script from the ORA_RDBMS directory.
4. Create a second rollback segment in the SYSTEM tablespace. Refer to the Oracle7 Server Administrator’s Guide for more information on creating rollback segments.

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

```
SQL> ALTER ROLLBACK SEGMENT <name> ONLINE;
```
Chapter 8

Optimizing Oracle7

This chapter describes basic tuning activities that optimize the performance of the Oracle7 Server on OpenVMS. This chapter contains the following major sections:

- Tuning Memory Usage for the Oracle7 Server
- Tuning I/O for the Oracle7 Server
- Installing products in shared memory
- Reducing database fragmentation
Tuning Memory Usage for the Oracle7 Server

There are various ways of tuning your Oracle7 system to improve system performance. This section describes the following four ways:

• Adjusting INIT.ORA parameters
• Modifying OpenVMS process quotas
• Adjusting the SGA

Refer to the Oracle7 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 ORACLEINS5 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 OpenVMS Process Quotas

At instance startup time, OpenVMS process quota limits are set for the Oracle7 Server detached (background) processes. 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 Oracle7 Server Sets Process Quotas

The Oracle7 Server automatically sets and adjusts the 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 B – 1 in Appendix B of this guide.

Warnings about Modifying the Process Quotas

If you do need to increase the OpenVMS quota limits, you can use the Oracle7 process quota logical names to do so. If you plan to change a process quota other than ENQLM, first consult Oracle Worldwide Technical Support. If you have insufficient OpenVMS quota limits, you will receive the Oracle7 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 Oracle7 account (or the account from which you will restart the instance).
2. Shut down the instance if it is currently running.
3. Define the 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 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 ORA_PROD_PQL$WSQUOTA  4096
```

If neither logical name is defined, Oracle7 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 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 OpenVMS memory space for the SGA. The size of the SGA and the SGA buffers is displayed whenever you start an Oracle7 instance as in the following example:
Total System Global Area 4787932 bytes
Fixed Size 45804 bytes
Variable Size 4611056 bytes
Database Buffers 122880 bytes
Redo Buffers 8192 bytes

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

Tuning I/O for the Oracle7 Server

For information about how the system is currently configured, you may issue the following command and get output as shown in Table 8–1. Then you can assess whether the most optimized path is being used.
SVRMGR> select * from v$sysstat where 2> statistic# > 144 and statistic# < 152;

<table>
<thead>
<tr>
<th>STATISTIC#</th>
<th>NAME</th>
<th>CLASS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>145</td>
<td>BUFFER OBJECTS</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>146</td>
<td>BUFFER OBJECT ALLOCATION FAILURES</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>147</td>
<td>TOTAL WRITE FANDLES</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>148</td>
<td>WRITE FANDLES IN USE</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>149</td>
<td>TOTAL READ FANDLES</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>150</td>
<td>READ FANDLES IN USE</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>151</td>
<td>FANDLE ALLOCATION FAILURES</td>
<td>16</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 8–1 Tuning I/O for the Oracle7 Server

where:

BUFFER OBJECTS are used to map memory that is used for performing OpenVMS Fast I/O. This includes I/O buffers and buffers used for I/O status. Each process normally uses two or three buffer objects. If the process has many I/Os simultaneously active to many files, more buffer objects may be in use.
BUFFER OBJECT ALLOCATION FAILURES should always be zero. If it is not zero, the process has been unable to create a buffer object. Then the server will fall back to using the $QIO system service instead of Fast I/O for some or all I/Os with some decline in performance. The possible causes are as follows:

- OpenVMS support for buffer objects in memory resident global sections is included in a post Alpha OpenVMS Version 7.1 kit. If this kit is not installed, buffer objects cannot be allocated.

  Additional Information: Refer to the Oracle7 for Alpha OpenVMS Release Notes, Version 7.3.4.3 Production for information about obtaining this post Version 7.1 kit.

- There may be resource limits that caused the failure. Check the OpenVMS Sysgen parameters MAXBOBMEM and GBLPAGES. The system may also be short of non–paged pool.

TOTAL WRITE FANDLES are preallocated Fast I/O packets. Total Write Fandles is the maximum number of simultaneously active write Fast I/Os.

WRITE FANDLES IN USE is the current number of active write Fast I/Os.

TOTAL READ FANDLES are preallocated Fast I/O packets. Total Read Fandles is the maximum number of simultaneously active read Fast I/Os.

READ FANDLES IN USE is the current number of active read Fast I/Os.

FANDLE ALLOCATION FAILURES should always be zero. If it is not zero, the process has been unable to create a preallocated Fast I/O packet. The server will fall back to using the QIO system service instead of Fast I/O for some or all I/Os with some decline in performance. The possible causes are as follows:

- OpenVMS support for buffer objects in memory resident global sections is included in a post Version 7.1 kit. If this kit is not installed, buffer objects cannot be allocated. Buffer objects are required for fandle creation.

  Additional Information: Refer to the Oracle7 for Alpha OpenVMS Release Notes, Version 7.3.4.3 Production for information about obtaining this post Version 7.1 kit.

- There may be resource limits that caused the failure. The system may also be short of non–paged pool.
If you want to get specifics for your session, issue the following command:

```
SVRMGR> select * from v$mystat where
  > statistic# > 144 and statistic# < 152
```

**Additional Information:** Please refer to Chapter 9 in the *OpenVMS I/O User’s Reference Manual, Version 7.1*. Note that this chapter states that buffer objects may not be created in global sections. However, the update with the post Version 7.1 kit allows buffer objects to be created in memory resident global sections.

### 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.

**Additional Information:** Refer to the *Oracle7 for Alpha OpenVMS Installation Guide, Version 7.3.4 Production* for information regarding Global Page Calculation.

#### 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 Oracle7 images.

Reducing Database Fragmentation

This section supplements the instructions for reducing database
fragmentation in the Oracle7 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 Oracle7 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 Oracle7 as SYSTEM and run the CATDBSYN.SQL file from the ORA_RDBMS directory.

9. Next, create a rollback segment in the SYSTEM tablespace. Refer to the *Oracle7 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:

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

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

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

11. Import the export file using the Import utility.
This chapter describes how to use Oracle7 Server 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 Oracle7 encounters an exception condition, such as an access violation or an attempt to divide a value by zero, Oracle7 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:

- OpenVMS process ID
- 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 SRV.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 Oracle7 Server 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

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

For example, a trace file created by process 005 running SQL*Plus against instance MKT1 might create a trace file called HARPO_MKT1_BG_SRV_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 Oracle7 Server Version 7.1.5 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
This chapter explains how to use the Oracle Programmatic Interfaces and SQL*Module in the 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
- Notes on using Pro*Ada
- SQL*Module
- Using event flags
Programmatic Interfaces

Note: 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*Ada
- Pro*C
- Pro*COBOL
- Pro*FORTRAN
- Pro*Pascal
- Pro*PL/I

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

ORA_ROOT

-- directory

PROGINT
Product Code
ORA_PROGINT

-- logical name

EXE
(*.CFG files, executables)
ORA_PROGINT_EXE
ORA_PROGINT_INCL

INCLUDE
(Precompiler include files)
ORA_PROGINT_INCLUDE
ORA_PROGINT_OCIALIB

OCI

MESG
(National lang. files)
ORA_PROGINT_MESG
ORA_PROGINT_SLAX

DEMO

OLB
(All object lib., object files, option files)
ORA_PROGINT_DEMO
ORA_PROGINT_OLB
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>
<thead>
<tr>
<th>Precompiler or SQL*Module</th>
<th>OpenVMS Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pro*Ada (version 1.8)</td>
<td>PROADA</td>
</tr>
<tr>
<td>Pro*C (version 2.2)</td>
<td>PROC</td>
</tr>
<tr>
<td>Pro*COBOL (version 1.8)</td>
<td>PROCOB</td>
</tr>
<tr>
<td>Pro*FORTRAN (version 1.8)</td>
<td>PROFOR</td>
</tr>
<tr>
<td>Pro*Pascal (version 1.6)</td>
<td>PROPAS</td>
</tr>
<tr>
<td>Pro*PL/I (version 1.6)</td>
<td>PROPLI</td>
</tr>
<tr>
<td>SQL*Module for C (version 1.1)</td>
<td>MODC</td>
</tr>
</tbody>
</table>

Table 10–1 OpenVMS Symbols

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.
- `<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.

The following guidelines and restrictions apply to precompiling.

**Using the OpenVMS Debugger**

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

**Using Event Flags**

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

**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:

```c
#define SQLCA_STORAGE_CLASS GLOBALDEF
```

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

```c
#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 Oracle7

Oracle7 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 5.7 compiler on 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 Oracle7 stack trace listings.

Floating Point Formats

Oracle7 is compiled with the default floating point format supported by the C compiler. Then the conversion routines within Oracle7 use this to translate operating system–specific floating point numbers into Oracle7 internal floating point representation.

Note: Oracle precompiler and OCI programmers should take special note that Oracle7 Alpha OpenVMS is compiled to recognize the G_FLOAT floating point format and Oracle7 VAX OpenVMS is compiled to use the D_FLOAT floating point format.

Pro*Ada, Pro*PL/I, or Pro*Pascal

When you compile any Pro*Ada, Pro*PL/I, or Pro*Pascal program with the appropriate Compaq compiler, you might get the following message:

Variable ABCDEFGH is fetched, not initialized

If this happens, you will need to compile your module with the /NOOPTIMIZE qualifier for it to work properly. Oracle is working with Compaq to eliminate this requirement in the future.
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 LOUTL.COM under special circumstances when LNPRO<language>.COM is not appropriate. If you decide to use LOUTL.COM, use a command syntax similar to that found in the appropriate LNPRO<language>.COM script.

**Note:** Due to architectural changes in the Oracle7 Server, user applications can no longer be linked directly against the Oracle shareable image (linking with the LOUTL “S” option). Single–task connections to the Oracle database are no longer supported.

All Oracle third party and user tools must now link against Oracle with the LOUTL “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 DECnet or SQL*Net TCP/IP adapters.

**Syntax**

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

```
$ LNPRO<language> <executable> <objectfilelist> <options>
```

where:

<language>    Abbreviation for the programming language you are using (ADA, C, COB, FOR, PAS, PLI).
<executable>  Name of the executable image to be created; a filename extension is optional.
<objectfilelist>  Comma-separated list of object files and libraries.
If this list is longer than one line, use the continuation character,–. Note that there are no spaces in this specification.
/options>  List of options with no separators needed:

D  Links with the OpenVMS DEBUG utility.
F  Produces a full map.
M  Creates a link map.
X  Produces a link map with cross references.

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 Ada executable called MYFILE (and to specify options D, and M), use the following command:

$ LNPROADA MYFILE MYOBJ,SUB DM

Including Option Files

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

For example, the following statement is incorrect:

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

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:

Reduced Executable Size by Using Shared Core and Shared UPI

The size of executables are reduced by using shared core and shared UPI. The links will default to using the shared libraries (if they exist.)

To avoid using shared libraries, you will need to define the following symbols to any non-null value:

- NO_SHARED_CORE_LIB
- NO_SHARED_UPI_LIB

The SHARED_CORE_LIB and SHARED_UPI_LIB symbols cause two new shareable images to be used during linking. They are called:

CORE<imageid>.EXE
UPISHR<imageid>.EXE

where <imageid> is the same as the identifier appended to the end of the Oracle shared image. These are found in the directory specified by the ORA_RDBMS logical name.

Optional Use of Shared SQLLIB

You can further reduce the size of the executables and increase performance by using shared SQLLIB. The shared image is called SQLLIBSHR<imageid>.EXE and is found in the directory specified by the ORA_UTIL logical name. In the small sample programs, using shared SQLLIB reduced the size of executables more than 10 times. Shared SQLLIB is automatically built by ORACLEINS, but is not used until you set the symbol in your link scripts.

If you are using the LNPRO<language> scripts and you want to use shared SQLLIB, do the following:

1. Set the following symbol to “yes” in your link scripts:
   \$ SHARED_SQL_LIB = "YES"

2. Make sure your program does not link against UTILCA. (Delete or comment out the line that links against UTILCA in your link script.)

3. Link your programs with the LNPRO<language> symbol.

If you are using the LOUTL link scripts and you want to use shared SQLLIB, do the following:
1. Replace the line used to link against the SQLLIB library to link against the SQLLIB shareable image.
   
   For example, to link against the SQLLIB library:
   
   ORA_UTIL:SQLLIB/LIB
   
   replace with a link against the SQLLIB shareable image:
   
   ORA_UTIL:SQLLIB/OPT
   
2. Make sure your program does not link against UTILCA. (Delete or comment out the line that links against UTILCA in your link script.)

3. Link your programs with the LOUTL.COM symbol.

   To return to non–shared SQLLIB, comment out the symbol
   
   SHARED_SQL_LIB and link your program again.

   You can remove shared SQLLIB by using the symbol REMSQLLIB. REMSQLLIB is a symbol for "@ORA_UTIL:REMSSQLLIB.COM." It executes a command file to remove the shared SQLLIB.

   You can reinstate shared SQLLIB by using the symbol INSSQLLIB. INSSQLLIB is a symbol for "@ORA_UTIL:INSSQLLIB.COM." It executes a command file to install shared SQLLIB.

   Note: If you reboot your system, be sure to invoke INSSQLLIB to keep using shared SQLLIB.

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.

**SQL*Module for C Demo**

Use the BLDCDEMO to run the SQL*Module for C demo.

   Note: SAMPLE5 is a user exit sample program, but user exits are not linked using LNPRO<language>.

   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.

Do Not Use ORA_RDBMS in Your Link Scripts

To improve client-only product linking, the following libraries, which were formerly in ORA_RDBMS, are now in ORA_UTIL:

- CLLIB.OLB
- CORELIB.OLB
- KLIB.OLB
- KUSR.OLB
- LLIB.OLB
- NLSRTL.OLB
- OCI.OLB

Linking Shareable Images with LOUTL.COM

You may link a shareable image against Oracle7 code using the LOUTL I and D options with LOUTL.COM or one of the LNPRO*.COM link scripts that internally calls LOUTL.COM. To eliminate missing file errors, also use the LOUTL flag LS (for Link Shareable).

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

```
$ INSTALL CREATE/SHARE/WRITE/HEADER <shareable_image>
```

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

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

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

Valid examples:

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

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

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, COBOL, and Ada are supported on OpenVMS for OCI programs.
OCI sample programs are supplied in the following directory:

ORA_ROOT: [RDBMS.DEMO.OCI_DEMO]

Guidelines

The following guidelines apply to using OCI routines:

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

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 sizes and offsets of the data elements differ between Alpha and VAX. This is because the sizes and offsets depend on the size of the native data types and the way the C compiler treats the post-compile layout of the structures in memory.

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>
<thead>
<tr>
<th>Structure Element</th>
<th>Offset (Bytes)</th>
<th>Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v2_rc</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>ft</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>rpc</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>peo</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>fc</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>rc</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>wrn</td>
<td>14</td>
<td>1</td>
</tr>
</tbody>
</table>
### Oracle Programmatic Interfaces

#### VAX 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 10–3.

<table>
<thead>
<tr>
<th>Structure Element</th>
<th>Offset (Bytes)</th>
<th>Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v2_rc</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>ft</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>rpc</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>peo</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>fc</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>rc</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>wrn</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>rid</td>
<td>20</td>
<td>13</td>
</tr>
<tr>
<td>ose</td>
<td>33</td>
<td>4</td>
</tr>
<tr>
<td>rcsp</td>
<td>38</td>
<td>4</td>
</tr>
</tbody>
</table>

*Table 10–3 VAX Size and Offset of Structure Elements*

**Note:** The alignment and padding differs from Alpha.

#### Linking OCI Programs Written in C

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

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

where:

- `<executable>` Name of the executable image to be created; a filename extension is not required.
- `<objfilelist>` 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:

D Links with the OpenVMS DEBUG utility.
F Produces a full map.
M Creates a link map.
X Produces a link map with cross references.

For example:
$ @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, COBOL, and Ada are supported on 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 Oracle7 Server.

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

Binary Integers

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

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Usage of Binary Integers</th>
<th>Usage of Short Binary Integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>int;</td>
<td>short;</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>INTEGER*4</td>
<td>INTEGER*2</td>
</tr>
<tr>
<td>COBOL</td>
<td>PIC S9(9) COMP</td>
<td>PIC S9(4) COMP</td>
</tr>
</tbody>
</table>

Table 10–4 Usage of Binary and Short Binary Integers

Using Literals as Call Arguments

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

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

Optional or Missing Parameters

In 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.
```

---

**Notes on Using Pro*Ada**

This section gives information about Pro*Ada.

**Post Pro*Ada Install**

After installing Pro*Ada, you must perform the following tasks:

1. If you do not have an Ada library or if you do not have a library created with the option `/LONG_FLOAT=D_FLOAT`, then create a new library by typing:

   ```
   $ ACS CREATE LIBRARY/LONG_FLOAT=G_FLOAT < directory >
   ```

   where `<directory>` is the directory where your Ada library file, ADALIB.ALB, will be located.

2. Reference this library by typing:

   ```
   $ ACS SET LIBRARY < directory >
   ```

3. Enter Oracle’s general library packages into your own library. For precompiler interfaces, type:

   ```
   $ ACS ENTER UNIT/REPLACE ORA_PROADALIB:[PROADALIB] *
   ```

   For OCI interfaces, type:

   ```
   $ ACS ENTER UNIT/REPLACE ORA_OCIADALIB:[OCIADALIB] *
   ```
4. Compile your Ada code.

5. Export your code from the ACS library.
   $ ACS EXPORT/MAIN <module>

6. Link your code using:
   $ LNPROADA <executable> <module>
   
   For OCI interfaces, type:
   $ LNOCI <executable> <module>

Relinking Pro*Ada

Starting with V1.5.10 of the Oracle Precompilers, the Pro*Ada runtime library gets compiled each time you link Pro*Ada on a system. In previous versions, the runtime library was compiled only the first time Pro*Ada was linked.

This change means that if you link the product more than once, you will not have to reload the entire product from the saveset to recompile the Pro*Ada runtime library. However, you may notice slightly longer PROGINT build times as a result of this change.

SQL*Module

SQL*Module is a development tool that enables an application to access data in an Oracle database. SQL*Module is currently available for the C language on OpenVMS.

SQL*Module enables an application developed in C to access relational databases without using embedded SQL and without using a proprietary application programming interface (API). In other words, with SQL*Module, the programmer can keep code for database operations separate from other code because all SQL programming can be done in a separately maintained module.

Using LNPRO<language> to Link Object and Module Files

You can use the LNPRO<language> to link object files, SQL*Module files, and precompiled files.

SQL*Module for C Demo

Use the BLDCDEMO to run the SQL*Module for C demo.
Using Event Flags

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

Oracle7 prevents asynchronous events from interfering with synchronous events by overwriting their event flags. This may increase the reliability of Oracle7 software on modern hardware, but it may introduce some problems for application programmers.

Oracle7 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 Oracle7. Event flags 1 and 5 are used by all synchronous events within Oracle7 and can also be used by application programmers. SYS$GETEF() is not used for these event flags.

SQL*Net Version 2 also uses additional event flags, which it gets dynamically from SYS$GETEF() calls from the second event flag group that ranges from 32–63. Make sure that you check the availability of any event flags you use in this range.

**Note:** Record Management Services (RMS) uses event flags 27 through 31.
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 in the 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:HELPINS

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). 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 TESPAR.SQL from the DCL command line, enter:

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

Within the TESPAR 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 the same as 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:

- GLOGIN.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  CPU:  0:00:00.19  BUFIO:  14
         DIRIO:  2  FAULTS:  23

where:

ELAPSED                Elapsed execution time, in seconds
CPU                    Total CPU time, in seconds
BUFIO                  Buffered I/O, in blocks
DIRIO                  Direct I/O, in blocks (1 block = 512 bytes)
FAULTS                 Number of page faults that occurred during execution

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 Compaq reference DEC C Runtime Library Functions and Macros.
PART III

Appendices
Topics covered in this appendix are:

- Overview of National Language Support
- Supported languages and character sets
- Overriding the default character sets and other attributes
- Supported product translations
- Defining linguistic sort sequences
- Message files
Overview of National Language Support

Oracle7 uses an integrated national-language architecture that supports national languages and character encoding. More than one language can be used by the same application.

National Language Support can be added without requiring a new release of Oracle software, and the language-dependent features can be customized for a specific application, user, or organization.

Applications may use different computer platforms for Oracle7 and application software. If the platforms use different character-encoding schemes, character data passed between client and server must be converted. Oracle7 makes this process transparent to the client application.

Supported Languages and Character Sets

The following tables lists the languages and character sets supported in this release.

Supported Languages

<table>
<thead>
<tr>
<th>Language Name</th>
<th>Oracle Language Specifier</th>
<th>Territory Name</th>
<th>Oracle Territory Specifier</th>
<th>Character Set</th>
<th>Language Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>American</td>
<td>AMERICAN</td>
<td>United States</td>
<td>AMERICA</td>
<td>US7ASCII</td>
<td>us</td>
</tr>
<tr>
<td>Arabic</td>
<td>ARABIC</td>
<td>United Arab Emirates</td>
<td>&quot;UNITED ARAB EMIRATES&quot;</td>
<td>AR8ISO8859P6</td>
<td>ar</td>
</tr>
<tr>
<td>Brazilian Portuguese</td>
<td>&quot;BRAZILIAN PORTUGUESE&quot;</td>
<td>Brazil</td>
<td>BRAZIL</td>
<td>WE8DEC</td>
<td>ptb</td>
</tr>
<tr>
<td>Czech</td>
<td>CZECH</td>
<td>Czech Republic</td>
<td>CZECHOSLOVAKIA</td>
<td>EE8ISO8859P2</td>
<td>cs</td>
</tr>
<tr>
<td>Danish</td>
<td>DANISH</td>
<td>Denmark</td>
<td>DENMARK</td>
<td>WE8DEC</td>
<td>dk</td>
</tr>
<tr>
<td>Dutch</td>
<td>DUTCH</td>
<td>The Netherlands</td>
<td>&quot;THE NETHERLANDS&quot;</td>
<td>WE8DEC</td>
<td>nl</td>
</tr>
<tr>
<td>Finnish</td>
<td>FINNISH</td>
<td>Finland</td>
<td>FINLAND</td>
<td>WE8DEC</td>
<td>sf</td>
</tr>
<tr>
<td>French</td>
<td>FRENCH</td>
<td>France</td>
<td>FRANCE</td>
<td>WE8DEC</td>
<td>f</td>
</tr>
<tr>
<td>German</td>
<td>GERMAN</td>
<td>Germany</td>
<td>GERMANY</td>
<td>WE8DEC</td>
<td>d</td>
</tr>
<tr>
<td>Greek</td>
<td>GREEK</td>
<td>Greece</td>
<td>GREECE</td>
<td>EL8DEC</td>
<td>el</td>
</tr>
<tr>
<td>Hungarian</td>
<td>HUNGARIAN</td>
<td>Hungary</td>
<td>HUNGARY</td>
<td>EE8ISO8859P2</td>
<td>hu</td>
</tr>
</tbody>
</table>
Table A – 1  Supported Languages, Territories, Character Sets, and Language Codes

<table>
<thead>
<tr>
<th>Language</th>
<th>Code</th>
<th>Territory</th>
<th>Code</th>
<th>Character Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Italian</td>
<td>ITALIAN</td>
<td>Italy</td>
<td>ITALY</td>
<td>WEB8DEC</td>
</tr>
<tr>
<td>Japanese</td>
<td>JAPANESE</td>
<td>Japan</td>
<td>JAPAN</td>
<td>JA16EUC</td>
</tr>
<tr>
<td>Korean</td>
<td>KOREAN</td>
<td>Korea</td>
<td>KOREA</td>
<td>KO16KSC5601</td>
</tr>
<tr>
<td>Norwegian</td>
<td>NORWEGIAN</td>
<td>Norway</td>
<td>NORWAY</td>
<td>WEB8DEC</td>
</tr>
<tr>
<td>Polish</td>
<td>POLISH</td>
<td>Poland</td>
<td>POLAND</td>
<td>EE8ISO8859P2</td>
</tr>
<tr>
<td>Portuguese</td>
<td>PORTUGUESE</td>
<td>Portugal</td>
<td>PORTUGAL</td>
<td>WEB8DEC</td>
</tr>
<tr>
<td>Russian</td>
<td>RUSSIAN</td>
<td>CIS</td>
<td>CIS</td>
<td>CL8ISO8859P5</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>&quot;SIMPLIFIED CHINESE&quot;</td>
<td>China</td>
<td>CHINA</td>
<td>ZHS16CGB231280</td>
</tr>
<tr>
<td>Slovak</td>
<td>SLOVAK</td>
<td>Slovakia</td>
<td>SLOVAKIA</td>
<td>EE8ISO8859P2</td>
</tr>
<tr>
<td>Spanish</td>
<td>SPANISH</td>
<td>Spain</td>
<td>SPAIN</td>
<td>WEB8DEC</td>
</tr>
<tr>
<td>Swedish</td>
<td>SWEDISH</td>
<td>Sweden</td>
<td>SWEDEN</td>
<td>WEB8DEC</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>&quot;TRADITIONAL CHINESE&quot;</td>
<td>China</td>
<td>ZHP</td>
<td>ZHT32EUC</td>
</tr>
<tr>
<td>Turkish</td>
<td>TURKISH</td>
<td>Turkey</td>
<td>TURKEY</td>
<td>WEB8ISO8859P9</td>
</tr>
</tbody>
</table>

Supported Character Sets

Oracle National Language Support supports the following character sets:

7-Bit Character Sets

- **AR7ASMO449PLUS**: Arabic/Latin ASMO-Plus 7-bit
- **CH7DEC**: DEC Swiss 7-bit
- **D7DEC**: DEC German 7-bit
- **E7DEC**: DEC Spanish 7-bit
- **F7DEC**: DEC French 7-bit
- **I7DEC**: DEC Italian 7-bit
- **NDK7DEC**: DEC Norwegian/Danish 7-bit
- **NL7DEC**: DEC Dutch 7-bit
- **SF7ASCII**: Finnish 7-bit ASCII extension
- **SF7DEC**: DEC Finnish 7-bit
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S7DEC</td>
<td>DEC Swedish 7-bit</td>
</tr>
<tr>
<td>TR7DEC</td>
<td>DEC Turkish 7-bit</td>
</tr>
<tr>
<td>US7ASCII</td>
<td>U.S. 7-bit ASCII (default)</td>
</tr>
</tbody>
</table>

8-bit Character Sets

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR8ASMO708PLUS</td>
<td>Arabic/Latin ASMO-Plus 8-bit</td>
</tr>
<tr>
<td>AR8ISO8859P6</td>
<td>ISO 8859–6 Latin / Arabic</td>
</tr>
<tr>
<td>CL8ISO8859P5</td>
<td>ISO 8859–5 Cyrillic 8-bit</td>
</tr>
<tr>
<td>CL8MACCYRILLIC</td>
<td>Mac Cyrillic 8-bit</td>
</tr>
<tr>
<td>CL8MSWINDOW31</td>
<td>Windows 3.1 Cyrillic 8-bit</td>
</tr>
<tr>
<td>DK8EBCDIC277</td>
<td>EBCDIC 8-bit Danish – code page 277</td>
</tr>
<tr>
<td>D8EBCDIC273</td>
<td>EBCDIC 8-bit Austrian/German – code page 273/1</td>
</tr>
<tr>
<td>EEC8EUROPA3</td>
<td>EEC’s EUROPA3 West European/Greek 8-bit character set</td>
</tr>
<tr>
<td>EE8ISO8859P2</td>
<td>ISO 8859–2 East European 8-bit</td>
</tr>
<tr>
<td>EE8PC852</td>
<td>IBM PC 8-bit East European – code page 852</td>
</tr>
<tr>
<td>EL8DEC</td>
<td>DEC Latin/Greek 8-bit</td>
</tr>
<tr>
<td>EL8EBCDIC875</td>
<td>EBCDIC 8-bit Greek – code page 875</td>
</tr>
<tr>
<td>EL8ISO8859P7</td>
<td>ISO 8859–7 Latin/Greek 8-bit</td>
</tr>
<tr>
<td>EL8PC437S</td>
<td>IBM-PC Special American/Greek character set</td>
</tr>
<tr>
<td>IW8ISO8859P8</td>
<td>ISO 8859–8 Latin/Hebrew (Iwriet) 8-bit</td>
</tr>
<tr>
<td>I8EBCDIC280</td>
<td>EBCDIC 8-bit Italian – code page 280/1</td>
</tr>
<tr>
<td>LT8PC772</td>
<td>IBM PC 8-bit Lithuanian – code page 772</td>
</tr>
<tr>
<td>LT8PC774</td>
<td>IBM PC 8-bit Lithuanian – code page 774</td>
</tr>
<tr>
<td>NEE8ISO8859P4</td>
<td>ISO 8859–4 North and North-east European</td>
</tr>
<tr>
<td>N8PC865</td>
<td>IBM PC 8-bit Norwegian – code page 865</td>
</tr>
<tr>
<td>RU8BESTA</td>
<td>Latin/Cyrillic BESTA 8-bit</td>
</tr>
<tr>
<td>RU8PC855</td>
<td>IBM-PC Latin/Cyrillic 8-bit – code page 855</td>
</tr>
<tr>
<td>RU8PC866</td>
<td>IBM-PC Latin/Cyrillic 8-bit – code page 866</td>
</tr>
<tr>
<td>SE8ISO8859P3</td>
<td>ISO 8859–3 South European 8-bit</td>
</tr>
<tr>
<td>TH8TISASCII</td>
<td>Thai Industrial Standard 620–2533 ASCII 8-bit</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TH8TISEBCDIC</td>
<td>Thai Industrial Standard 620–2533 EBCDIC 8-bit</td>
</tr>
<tr>
<td>TR8DEC</td>
<td>DEC Turkish 8-bit</td>
</tr>
<tr>
<td>TR8ISO8859P9</td>
<td>Turkish version ISO 8859–9 West European</td>
</tr>
<tr>
<td>TR8PC857</td>
<td>IBM-PC Turkish 8-bit – code page 857</td>
</tr>
<tr>
<td>US8ICL</td>
<td>ICL EBCDIC 8-bit US</td>
</tr>
<tr>
<td>US8PC437</td>
<td>IBM PC 8–bit U.S. – code page 437</td>
</tr>
<tr>
<td>WE8DEC</td>
<td>DEC West European 8-bit</td>
</tr>
<tr>
<td>WE8EBCDIC37</td>
<td>EBCDIC 8-bit West European – code page 37</td>
</tr>
<tr>
<td>WE8EBCDIC500</td>
<td>EBCDIC 8-bit West European – code page 500</td>
</tr>
<tr>
<td>WE8HP</td>
<td>HP 8-bit West European</td>
</tr>
<tr>
<td>WE8ICL</td>
<td>ICL EBCDIC 8-bit West European</td>
</tr>
<tr>
<td>WE8ISO8859P1</td>
<td>ISO 8859–1 West European 8-bit</td>
</tr>
<tr>
<td>WE8ISO8859P9</td>
<td>ISO 8859–9 West European/Turkish 8-bit</td>
</tr>
<tr>
<td>WE8PC850</td>
<td>IBM PC 8-bit West European – code page 850 (for use with HFT terminals)</td>
</tr>
</tbody>
</table>

**Multi-Byte Character Sets**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JA16TSTSET2</td>
<td>Japanese</td>
</tr>
<tr>
<td>JA16DBCS</td>
<td>Japanese IBM</td>
</tr>
<tr>
<td>JA16EBCDIC930</td>
<td>Japanese Extended UNIX Code</td>
</tr>
<tr>
<td>JA16SGS</td>
<td>Japanese Shift–JIS</td>
</tr>
<tr>
<td>JA16TSTSET</td>
<td>Japanese</td>
</tr>
<tr>
<td>JA16VMS</td>
<td>Japanese VMS Kanji</td>
</tr>
<tr>
<td>KO16DBCS</td>
<td>Korean IBM</td>
</tr>
<tr>
<td>KO16KSC5601</td>
<td>Korean KSC5601</td>
</tr>
<tr>
<td>ZHS16CGB231280</td>
<td>Chinese GB2312–80</td>
</tr>
<tr>
<td>ZHT16BIG5</td>
<td>BIG5 Traditional Chinese</td>
</tr>
</tbody>
</table>
Overriding the Default Character Sets and Other Attributes

When you create a database, you specify the character set in which the data is stored. You cannot change the way the data is stored later.

For example, if you create a database with a 7-bit character set, and users later enter data using an 8-bit character set, the database will not be able to store the extra bit. Therefore, when you create a database, specify a character set that will accommodate all your users. Table A–1 lists all the character sets available in this release.

While you cannot change the way data is stored, there are linguistic attributes that you can specify and later override. In the following list, each method overrides the values set by the previous method.

1. Default characteristics
2. Initialization file parameters
3. Logical names
4. ALTER SESSION parameters

Default Characteristics

By default, the Oracle Server uses American and America for language and territory, respectively.

Initialization File Parameters

To use a language other than American English for a particular instance, set the NLS_LANGUAGE and NLS_TERRITORY parameters in the initialization parameter file for that instance.

Logical Names

Once an instance is started, you can override its language, territory, and associated linguistic characteristics by using the logical names in Table A–2.
### Table A–2 Additional Parameters and Logical Names

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLS_DATE_FORMAT</td>
<td>Default date format; for example, “DD/MON/YY”</td>
</tr>
<tr>
<td>NLS_DATE_LANGUAGE</td>
<td>Language for day and month names; for example, “German”</td>
</tr>
<tr>
<td>NLS_NUMERIC_CHARACTERS</td>
<td>Decimal character and group separator; for example, “.”</td>
</tr>
<tr>
<td>NLS_CURRENCY</td>
<td>Local currency symbol; for example, “Dfl”</td>
</tr>
<tr>
<td>NLS_ISO_CURRENCY</td>
<td>ISO currency symbol; for example, “France”</td>
</tr>
<tr>
<td>NLS_SORT</td>
<td>Sort sequence; for example, “Swedish”</td>
</tr>
</tbody>
</table>

### ALTER SESSION Parameters

Once an instance is started, a user with the ALTER SESSION privilege can use the SQL command ALTER SESSION to specify new values for his or her own session. Doing so overrides the values that have been specified in either the initialization parameter file or through the use of logical names.

To see the current values for your session, enter the following command from SQL*Plus:

```
SQL> SELECT * FROM NLS_SESSION_PARAMETERS;
```

**Additional Information:** To find out the ALTER SESSION parameters, refer to the *Oracle7 Server Administrator’s Guide*.

---

### Supported Product Translations

Table A–3 lists supported product translations.
Table A – 3 Supported Product Translations

<table>
<thead>
<tr>
<th>Product</th>
<th>AE</th>
<th>9WE</th>
<th>4E</th>
<th>5CE</th>
<th>A</th>
<th>3A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle7 Server</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Oracle TRACE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PL/SQL version 2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SQL*Net version 2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pro*Ada version 1.8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pro*C version 2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pro*COBOL version 1.8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pro*FORTRAN version 1.8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pro*Pascal version 1.6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pro*PL/1 version 1.6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SQL*Module</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Server Manager version 2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

In the table headings:

- **AE**: American English
- **9WE**: Danish, Dutch, Finnish, French, German, Italian, Norwegian, Spanish, and Swedish
- **4E**: Portuguese, Brazilian Portuguese, Greek, and Turkish
- **5CE**: Czech, Hungarian, Polish, Slovak, and Russian
- **A**: Arabic
- **3A**: Japanese, Korean, and Simplified Chinese

The following products have message files and help text for American English only:

- Oracle DECnet Protocol Adapter
- VMS Oracle Installer (ORACLEINS)
- Oracle TCP/IP Protocol Adapter

### Defining Linguistic Sort Sequences

Oracle7 provides linguistic sort sequences, enabling text to be sorted according to specific linguistic conventions. For example, “a umlaut” can be sorted before “b” in German, but after “z” in Swedish. Linguistic
sort sequences are defined by name, and specified with the NLS_SORT parameter using the syntax:

\[
\text{NLS\_SORT} = \text{BINARY | name}
\]

where \( \text{name} \) is the name of a linguistic sort sequence. In most cases, the value for \( \text{name} \) is identical to the corresponding language.

Syntax Example

\[
\begin{align*}
\text{NLS\_SORT} &= \text{BINARY} \\
\text{NLS\_SORT} &= \text{"brazilian portuguese"} \\
\text{NLS\_SORT} &= \text{German}
\end{align*}
\]

The sort mechanism used by an instance (binary or linguistic) can be changed by restarting the instance with a different value of NLS_SORT in the \(<\text{nodename}>_<\text{sid}>_{\text{INIT.ORA}}\) file.

Supported Sort Sequences

The linguistic sort sequence also specifies how to handle special cases in sorting and case conversion. For example, the “XSpanish” linguistic sort sequence treats the double characters “CH” and “LL” as single characters.

Available linguistic sort sequences are:

<table>
<thead>
<tr>
<th>Arabic</th>
<th>EEC_Euro</th>
<th>Italian</th>
<th>Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bengali</td>
<td>Estonian</td>
<td>Japanese</td>
<td>XSpanish</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>Finnish</td>
<td>Latin</td>
<td>Swedish</td>
</tr>
<tr>
<td>Canadian French</td>
<td>French</td>
<td>Latvian</td>
<td>Swiss</td>
</tr>
<tr>
<td>Catalan</td>
<td>XFrench</td>
<td>Lithuanian</td>
<td>XSwiss</td>
</tr>
<tr>
<td>XCatalan</td>
<td>German</td>
<td>Malay</td>
<td>Thai_Dictionary</td>
</tr>
<tr>
<td>Croatian</td>
<td>XGerman</td>
<td>Norwegian</td>
<td>Thai_Telephone</td>
</tr>
<tr>
<td>XCroatian</td>
<td>German_Din</td>
<td>Polish</td>
<td>Turkish</td>
</tr>
<tr>
<td>Czech</td>
<td>XGerman_Din</td>
<td>Romanian</td>
<td>XTurkish</td>
</tr>
<tr>
<td>XCzech</td>
<td>Greek</td>
<td>Russian</td>
<td>Ukranian</td>
</tr>
<tr>
<td>Danish</td>
<td>Hebrew</td>
<td>Slovak</td>
<td>Vietnamese</td>
</tr>
<tr>
<td>XDanish</td>
<td>Hungarian</td>
<td>XSlovak</td>
<td>West_European</td>
</tr>
<tr>
<td>Dutch</td>
<td>XHungarian</td>
<td>Slovenian</td>
<td>XWest_European</td>
</tr>
<tr>
<td>XDutch</td>
<td>Icelandic</td>
<td>XSlovenian</td>
<td></td>
</tr>
</tbody>
</table>
Message Files

The value you specify for LANGUAGE or NLS_LANG determines which language messages appear in. Each language has a set of message files, one per product.

Location of Message Files

On OpenVMS, the binary–format message files (those with the .MSB extension) are stored in the ORA_NLS directory.

This directory contains only the message files for the languages you have not installed. Once you install a language, its message files get deleted from this directory and move to the product directories of the installed products.

Deleting Message Files

To reclaim disk space, you can delete unneeded message files from ORA_NLS. The names of these files are in the following format:

<product code><language code>.MSB

where language code is one of the codes in the last column of Table A – 1.

For example, if you don’t need messages for any product to appear in Finnish, you can issue the following command from within ORA_NLS:

$ DELETE *SF.MSB

Caution: When using wildcards, be careful not to delete other languages that match the wildcard.
This appendix provides information about Oracle7 logical names and utilities, and the default and recommended values for various initialization parameters. Refer to the Oracle7 Server Administrator’s Guide for general information about all the initialization parameters.

This appendix contains the following major sections:

- Oracle7 logical names
- System–dependent initialization parameters
Oracle7 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 Oracle7 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 Worldwide Technical Support does not support modification of any command file, symbol, or logical name.
used by the Oracle7 Server 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 Oracle7 logical names. They are presented in alphabetical order.

**ORA_CONTROL1**

Identifies the location and name of the control file associated with the database that is currently open. ORA_CONTROL1 is defined by the ORA_DB:ORA_DB_<dbname>.COM file.

**ORA_CONTROL2**

Identifies the location and name of the second control file associated with the database that is currently open. ORA_CONTROL2 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 page 11 – 8 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 Oracle7 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 Oracle7 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 Oracle7 database is created. This script sets up the core of the Oracle7 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 Oracle7 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 Oracle7. 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 page 11–8 for more information.

ORA_PLS and ORA_PLSQL Identifies the message files for PL/SQL.

ORA_RDBMS Identifies the directory where the Oracle7 Server is installed. This directory contains the Oracle7 Server object libraries and the command files that build and link the shareable Oracle7 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 Oracle7 system, usually a subdirectory of the Oracle7 account’s login directory. ORA_ROOT is defined directly by the ORA_UTIL:ORAUSER.COM file.

ORA_SID Identifies the current Oracle7 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 Oracle7 utilities, libraries, and data files referenced by many Oracle products reside. This directory also contains the ORAUSER.COM file that defines Oracle7 logical names and command symbols and is executed whenever Oracle7 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_PROGIN1, 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 V2 administration directories.

Process Quota Logical Names If you don’t set quotas for Oracle’s background processes, the Oracle7 Server uses its own formulas to determine how to set the quota logical names. Table B - 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>
<thead>
<tr>
<th>Quota Logical Name</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Current Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQL$_ASTLM</td>
<td>0</td>
<td>65536</td>
<td>(\text{max # of dispatcher connections + max # instances}) * (\text{max # processes/instance} + 64) and 64 larger than write batch size</td>
</tr>
<tr>
<td>PQL$_BIOLM</td>
<td>0</td>
<td>1024</td>
<td>(\text{max # of dispatcher connections + 64})</td>
</tr>
<tr>
<td>PQL$_BYTLM</td>
<td>1024</td>
<td>1048576</td>
<td>default</td>
</tr>
<tr>
<td>PQL$_CPULM</td>
<td>0</td>
<td>0</td>
<td>default</td>
</tr>
<tr>
<td>PQL$_DIOLM</td>
<td>0</td>
<td>2048</td>
<td>20 larger than write batch size</td>
</tr>
<tr>
<td>PQL$_ENQLM</td>
<td>256</td>
<td>32767</td>
<td>32000 if multi-instance</td>
</tr>
<tr>
<td>PQL$_FILLM</td>
<td>64</td>
<td>65535</td>
<td>(\text{max # of dispatcher connections + 64})</td>
</tr>
<tr>
<td>PQL$_JTQUOTA</td>
<td>0</td>
<td>0</td>
<td>default</td>
</tr>
<tr>
<td>PQL$_PGFLQUOTA</td>
<td>20480</td>
<td>4194304</td>
<td>((\text{SGA}_\text{SIZE} + \text{K1}_\text{SIZE})/512)</td>
</tr>
<tr>
<td>PQL$_PRCLM</td>
<td>0</td>
<td>20</td>
<td>default</td>
</tr>
<tr>
<td>PQL$_TQELM</td>
<td>10</td>
<td>2048</td>
<td>Number of background processes plus 20.</td>
</tr>
<tr>
<td>PQL$_WSDEFAULT</td>
<td>2048</td>
<td>100000</td>
<td>default</td>
</tr>
<tr>
<td>PQL$_WSEXTENT</td>
<td>2048</td>
<td>1048576</td>
<td>((\text{SGA}_\text{SIZE} + 0.6 \times \text{K2}_\text{SIZE})/512)</td>
</tr>
<tr>
<td>PQL$_WSQUOTA</td>
<td>2048</td>
<td>1048576</td>
<td>((\text{SGA}_\text{SIZE} + 0.3 \times \text{K2}_\text{SIZE})/512)</td>
</tr>
</tbody>
</table>

In Table B–1, the variables SGA\_SIZE, K1\_SIZE and K2\_SIZE represent the following:

- **SGA\_SIZE**: The size of the SGA.
- **K1\_SIZE**: Process private storage + room for expansion. Includes all writable data. This value is hard-coded to 28 Mb (56,000 pages)
- **K2\_SIZE**: Expected size of entire background server’s P0 address space. This value is hard-coded to 40 Mb (80,000 pages)

For more information about modifying Oracle process quotas through logical names, see 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

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Identifies the directory where the trace files created by the detached Oracle7 processes are sent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended Value</td>
<td>The directory identified by the ORA_DUMP logical name.</td>
</tr>
<tr>
<td>Default Value</td>
<td>ORA_DUMP. By default, the ORA_DUMP logical name is assigned to the [.TRACE] subdirectory of ORA_DB.</td>
</tr>
<tr>
<td>Distributed Value</td>
<td>None</td>
</tr>
</tbody>
</table>

## CONTROL_FILES

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Identifies the name and location of the database control files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended Value</td>
<td>The names of all control files created by the database that can be accessed by the current instance: ORA_CONTROL1, ORA_CONTROL2.</td>
</tr>
<tr>
<td>Default Value</td>
<td>ORA_CONTROL1</td>
</tr>
<tr>
<td>Distributed Value</td>
<td>ORA_CONTROL1, ORA_CONTROL2.</td>
</tr>
</tbody>
</table>

## DB_BLOCK_SIZE

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Identifies size (in bytes) of Oracle7 database blocks and the database buffers in the SGA.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended Value</td>
<td>A multiple of the OpenVMS I/O block size of 512 bytes. The maximum value is 32768.</td>
</tr>
<tr>
<td>Default Value</td>
<td>2048</td>
</tr>
<tr>
<td>Distributed Value</td>
<td>None</td>
</tr>
<tr>
<td>Parameter</td>
<td>Purpose</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOG_ARCHIVE_DEST</td>
<td>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.</td>
</tr>
<tr>
<td>LOG_ARCHIVE_FORMAT</td>
<td>Specifies a default filename format for archived redo log files. LOG_ARCHIVE_FORMAT is appended to the string specified in the LOG_ARCHIVE_DEST parameter. The redo log file format specifications can contain variables that are substituted with a unique archived redo log file name. Refer to the “Recovering a Database” chapter of the Oracle7 Server Administrator’s Guide for more information on these variables.</td>
</tr>
<tr>
<td>PRE_PAGE_SGA</td>
<td>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.</td>
</tr>
</tbody>
</table>
**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 OpenVMS, the minimum value is 3.5 Mb.

**Default Value**
3.5 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

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Determines whether to use a backing file for the SGA instead of using a memory resident global section.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended Value</td>
<td>FALSE, unless there is a well understood need to allow the SGA to page. Setting this parameter to TRUE will disable the use of OpenVMS Fast I/O, slow process startup, and in most cases reduce performance.</td>
</tr>
<tr>
<td>Default Value</td>
<td>FALSE</td>
</tr>
<tr>
<td>Distributed Value</td>
<td>None</td>
</tr>
</tbody>
</table>
This appendix lists the Oracle7 messages and codes that are specific to the OpenVMS environment. These messages and codes supplement those in the Oracle7 Server Messages. This chapter also contains messages and codes that are common in, yet not reserved specifically for, the OpenVMS environment.

The following are OpenVMS messages and generic messages that are particularly applicable to the Oracle7 Server on OpenVMS. All messages between 07500 and 07999 are OpenVMS operating system dependent. For more information on Oracle messages and codes, refer to the Oracle7 Server Messages.

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

**Cause**  This is an OpenVMS error message that occurs when you try to run an Oracle7 tool without installing the Oracle7 shareable image. The two most likely image names that will be listed are ORA_UPISHR<image id> and ORA_RDBMS<image id>.

**Action**  Install Oracle7 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 OpenVMS rights database
- whether the account has the process rights identifier ORA_DBA, ORA_<sid>_DBA, or both

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

- If the identifier, ORA_<sid_x>_DBA, exists in the 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 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 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–07505: scggt: $enq parent lock unexpected return

Cause: OpenVMS system service $ENQ returned an unexpected value.
Action: Check for a system error message and refer to OpenVMS system documentation.

ORA–07506: scgrl: $deq unexpected return

Cause: OpenVMS system service $DEQ returned an unexpected value.
Action: Check for a system error message and refer to 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 OpenVMS system documentation.

ORA–07508: scgfal: $deq all unexpected return

Cause: OpenVMS system service $DEQ returned an unexpected value.
Action: Check for a system error message and refer to OpenVMS system documentation.

ORA–07509: scgfal: $deq parent lock unexpected return

Cause: OpenVMS system service $DEQ returned an unexpected value.
Action: Check for a system error message and refer to OpenVMS system documentation.

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

Cause: The OpenVMS system service $GETLKI returned an unexpected value.
Action: Check for a system error message and refer to the OpenVMS system documentation.

ORA–07511: sscggtl: $ENQ unexpected return for master termination lock

Cause: The OpenVMS system service, $ENQ returned an unexpected value.
Action: Check for a system error message and refer to the OpenVMS system documentation.
ORA-07512: sscgctl: $ENQ unexpected return for client termination lock

Cause  The OpenVMS system service, $ENQ returned an unexpected value.
Action Check for a system error message and refer to the OpenVMS system documentation.

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

Cause  The OpenVMS system service, $DEQ returned an unexpected value.
Action Check for a system error message and refer to the OpenVMS system documentation.

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

Cause  The OpenVMS system service, $DEQ returned an unexpected value.
Action Check for a system error message and refer to the 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  OpenVMS system service $OPEN failed.
Action Check for a system error message and refer to 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 Worldwide Technical Support.
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**  
OpenVMS system service $CREATE failed.

**Action**  
Check for a system error message and refer to 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**  
OpenVMS system service $CONNECT failed.

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

ORA–07524: sfccf: $write (zero file) error

**Cause**  
OpenVMS system service $WRITE failed.

**Action**  
Check for a system error message and refer to OpenVMS system documentation.
ORA–07525: sfccf: $close error
Causes: OpenVMS system service $CLOSE failed.
Actions: Check for a system error message and refer to OpenVMS system documentation.

ORA–07526: sfifi: illegal logical block size
Causes: 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.
Actions: Change DB_BLOCK_SIZE in the parameter file to conform to these limits.

ORA–07527: sfifi: UIC group <= MAXSYSGROUP – file operations not allowed
Causes: 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.
Actions: 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
Causes: OpenVMS system service $OPEN failed.
Actions: Check for a system error message and refer to OpenVMS system documentation.

ORA–07529: sfifi: $close error
Causes: OpenVMS system service $CLOSE failed.
Actions: Check for a system error message and refer to OpenVMS system documentation.

ORA–07530: sfifi: $open error
Causes: OpenVMS system service $OPEN failed.
Actions: Check for a system error message and refer to OpenVMS system documentation.

ORA–07531: ssfccf: $DISPLAY error
Causes: OpenVMS system service $DISPLAY failed.
Actions: Examine system error message and refer to OpenVMS system documentation.
ORA–07532:  sfci:  $dassgn error

Cause  OpenVMS system service $DASSGN failed.
Action  Check for a system error message and refer to 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  OpenVMS system service $GETLKI returned an unexpected value.
Action  Check for a system error message and refer to OpenVMS system documentation.

ORA–07535:  srfb:  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 Worldwide Technical Support.

ORA–07536:  srfb:  $qio(read) error

Cause  OpenVMS system service $QIO failed.
Action  Check for a system error message and refer to 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 Worldwide Technical Support.

ORA–07539: sfwrt: $qio (write) error

Cause
OpenVMS system service $QIO failed.

Action
Check for a system error message and refer to 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 OpenVMS system documentation, or contact Oracle Worldwide Technical Support.


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 OpenVMS system documentation.

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

Cause 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 OpenVMS system service $OPEN failed.

Action Check for a system error message and refer to 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 Worldwide Technical Support.

ORA-07549: sftopn: $OPEN failure

Cause  OpenVMS system service $OPEN failed.
Action  Check for a system error message and refer to OpenVMS system documentation.

ORA-07550: sftopn: $CONNECT failure

Cause  OpenVMS system service $CONNECT failed.
Action  Check for a system error message and refer to OpenVMS system documentation.

ORA-07551: sftcls: $CLOSE failure

Cause  OpenVMS system service $CLOSE failed.
Action  Check for a system error message and refer to OpenVMS system documentation.

ORA-07552: sftget: $GET failure

Cause  OpenVMS system service $GET failed.
Action  Check for a system error message and refer to OpenVMS system documentation.

ORA-07553: sfofi: out of open files

Cause  The number of open files has exceeded a OpenVMS Oracle7 compile time limit.
Action  This is an internal error. Verify that you can reproduce the error and contact Oracle Worldwide Technical Support.

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 Worldwide Technical Support.
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 Worldwide Technical Support.

ORA–07556:  sfotf: $create error
  Cause  OpenVMS system service $CREATE failed.
  Action  Examine system error message and refer to 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  OpenVMS system service $CREATE failed
  Action  Examine system error message and refer to 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:  sltln: $trnlog error
  Cause  Translation of a logical name failed (for example, due to overflow, too many levels of logical names, or the logical name was not defined at all).
  Action  Define the logical name or look for a name like ORA_SID that is exceptionally long or defined circularly. If none, report as a bug.

ORA–07561:  szprv: $IDTOASC failure
  Cause  OpenVMS system service $IDTOASC failed
  Action  Examine system error message and refer to 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 Worldwide Technical Support.

ORA-07563:  sldext: $PARSE failure
  Cause  OpenVMS system service $PARSE failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA-07564:  sldext: wildcard in filename or extension
  Cause  A wildcard was used in the filename.
  Action Reenter the filename completely.

ORA-07565:  sldext: $SEARCH failure
  Cause  OpenVMS system service $SEARCH failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA-07568:  slspool: $OPEN failure
  Cause  OpenVMS system service $OPEN failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA-07569:  slspool: $CLOSE failure
  Cause  OpenVMS system service $CLOSE failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA-07570:  szrfc: $IDTOASC failure
  Cause  OpenVMS system service $IDTOASC failed.
  Action Check for a system error message and refer to OpenVMS system documentation.
ORA–07571: \textit{szrfc: }$\textit{FIND\_HELD}$ failure
\begin{itemize}
  \item \textbf{Cause} OpenVMS system service $\textit{IDTOASC}$ failed
  \item \textbf{Action} Check for a system error message and refer to OpenVMS system documentation.
\end{itemize}

ORA–07572: \textit{szrfc: }insufficient rolename buffer space
\begin{itemize}
  \item \textbf{Cause} An OS role name was too long.
  \item \textbf{Action} Redefine the role name to be of correct length.
\end{itemize}

ORA–07573: \textit{slkhst: }could not perform host operation
\begin{itemize}
  \item \textbf{Cause} OpenVMS system service LIB$\texttt{SPAWN}$ failed.
  \item \textbf{Action} Check for a system error message and refer to OpenVMS system documentation.
\end{itemize}

ORA–07574: \textit{szrfc: }$\textit{GETUAI}$ failure
\begin{itemize}
  \item \textbf{Cause} OpenVMS system service $\textit{GETUAI}$ failed.
  \item \textbf{Action} Check for a system error message and refer to OpenVMS system documentation.
\end{itemize}

ORA–07576: \textit{sspexst: }$\textit{GETJPIW}$ failure on process id %s
\begin{itemize}
  \item \textbf{Cause} OpenVMS system service $\textit{GETJPIW}$ failed.
  \item \textbf{Action} Check for a system error message and refer to OpenVMS system documentation.
\end{itemize}

ORA–07577: no such user in authorization file
\begin{itemize}
  \item \textbf{Cause} An attempt was made to set an INTERNAL password (for either DBA or OPER privilege), but the corresponding VMS account (either ORA_$<$\textit{sid}$>$\_DBA or ORA_$<$\textit{sid}$>$\_OPER) hasn’t been created yet.
  \item \textbf{Action} Add a VMS account for ORA_$<$\textit{sid}$>$\_DBA and/or ORA_$<$\textit{sid}$>$\_OPER before trying to set a password for them.
\end{itemize}

ORA–07578: \textit{szprv: }$\textit{FIND\_HELD}$ failure
\begin{itemize}
  \item \textbf{Cause} OpenVMS system service $\textit{FIND\_HELD}$ failed.
  \item \textbf{Action} Check for a system error message and refer to OpenVMS system documentation.
ORA-07579:  spini:  $DCLEXH failure  
Cause  OpenVMS system service $DCLEXH failed.  
Action  Check for a system error message and refer to OpenVMS system documentation.

ORA-07580:  spstp:  $GETJPIW failure  
Cause  OpenVMS system service $GETJPIW failed.  
Action  Check for a system error message and refer to 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 Worldwide Technical Support.

ORA-07582:  spstp:  SID has illegal value  
Cause  The SID must exist and be less than 6 characters.  
Action  Refer to the Oracle7 for OpenVMS Installation Guide for your platform 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  OpenVMS system service $PARSE failed.  
Action  Check for a system error message and refer to OpenVMS system documentation.
ORA–07586: spdcr: $SEARCH failure

**Cause**  
OpenVMS system service $SEARCH failed.

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

ORA–07587: spdcr: $CREPRC failure

**Cause**  
OpenVMS system service $CREPRC failed.

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

ORA–07588: spdcr: $GETJPIW get image name failure

**Cause**  
OpenVMS system service $GETJPIW failed.

**Action**  
Check for a system error message and refer to 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**  
OpenVMS system service $DELPRC failed.

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

ORA–07591: spdde: $GETJPIW failure

**Cause**  
OpenVMS system service $GETJPIW failed.

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

ORA–07592: sspgprv: error obtaining required privileges

**Cause**  
While obtaining needed privileges, an error was returned from SYS$SETPRV.

**Action**  
This is an internal error. Verify that you can reproduce the error and contact Oracle Worldwide Technical Support.
ORA–07593: ssprpv: error release privileges
  Cause  While releasing privileges, an error was returned from SYS$SETPRV.
  Action  This is an internal error. Verify that you can reproduce the error and contact Oracle Worldwide Technical Support.

ORA–07594: spiip: $GETJPIW failed
  Cause  OpenVMS system service $GETJPIW failed.
  Action  Check for system error message and refer to OpenVMS system documentation.

ORA–07595: sppid: $GETJPIW failure
  Cause  OpenVMS system service $GETJPIW failed.
  Action  Check for a system error message and refer to OpenVMS system documentation.

ORA–07596: sptpa: $GETJPIW failure
  Cause  OpenVMS system service $GETJPIW failed.
  Action  Check for a system error message and refer to OpenVMS system documentation.

ORA–07597: spguns: $GETJPIW failure
  Cause  OpenVMS system service $GETJPIW failed.
  Action  Check for a system error message and refer to OpenVMS system documentation.

ORA–07598: spwat: $SETIMR failure
  Cause  OpenVMS system service $SETIMR failed.
  Action  Check for a system error message and refer to OpenVMS system documentation.

ORA–07599: spwat: $HIBER failure
  Cause  OpenVMS system service $HIBER failed.
  Action  Check for a system error message and refer to OpenVMS system documentation.
ORA–07600: spwat: $CANTIM failure
  Cause  OpenVMS system service $CANTIM failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA–07601: spmguno: $GETJPIW failure
  Cause  OpenVMS system service $GETJPIW failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA–07602: spgto: $GETJPIW failure
  Cause  OpenVMS system service $GETJPIW failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA–07605: szprv: $PARSE_ACL failure
  Cause  OpenVMS system service $PARSE_ACL failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA–07606: szprv: $CHKPRO failure
  Cause  OpenVMS system service $CHKPRO failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA–07607: szaud: $SNDOPR failure
  Cause  OpenVMS system service $SNDOPR failed.
  Action Check for a system error message and refer to OpenVMS system documentation.

ORA–07608: szprv: $GETUAI failure
  Cause  OpenVMS system service $GETUAI failed.
  Action Check for a system error message and refer to OpenVMS system documentation.
ORA–07609: **szprv: $HASH_PASSWORD failure**

**Cause**  
OpenVMS system service $HASH_PASSWORD failed.

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

ORA–07610: **$GETJPIW failed in retrieving the user’s MAC privileges**

**Cause**  
OpenVMS system service $GETJPIW failed.

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

ORA–07612: **$GETUAI failed in retrieving the user’s clearance level**

**Cause**  
OpenVMS system service $GETUAI failed.

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

ORA–07613: **szlgtl: $ASCTOID failure**

**Cause**  
OpenVMS system service $ASCTOID failed.

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

ORA–07618: **$IDTOASC failed translating a secrecy level**

**Cause**  
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**  
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, OpenVMS system service $CREATE failed.

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

ORA–07623:  smscre: $CRMPSC failure

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

**Action**  Examine the system error message and refer to 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:

```sql
$ 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: $DBGLSC failure

**Cause** While deleting the system global area (SGA), OpenVMS system service $DBGLSC failed.

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

### ORA–07625: smsget: $MGBLSC failure

**Cause** While mapping the system global area (SGA) during logon, the OpenVMS system service $MGBLSC failed. The usual reason is that Oracle7 has not been started up.

**Action** Examine the system error message and refer to OpenVMS system documentation. Start up Oracle7 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 Worldwide Technical Support.

### ORA–07627: smsfre: $CRETV A failure

**Cause** While unmapping the system global area (SGA) during logoff, OpenVMS system service $CRETV A failed.

**Action** Examine the system error message and refer to 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 Worldwide Technical Support.

ORA–07629: smpall: $EXPREG failure

Cause While creating the program global area (PGA) during logon, 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 OpenVMS system documentation.

ORA–07630: smpdal: $DELTVA failure

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

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

ORA–07631: smcacx: $EXPREG failure

Cause While creating or extending a context area, 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 OpenVMS system documentation.

ORA–07632: smsrcx: $DELTVA failure

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

Action Examine the system error message and refer to 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 Worldwide Technical Support.
ORACLE 07634: smsdbp: $CRETV A failure

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

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

ORACLE 07635: smsdbp: $SETPRT failure

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

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

ORACLE 07636: smsdbp: $MGBLSC failure

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

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

ORACLE 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 Worldwide Technical Support.

ORACLE 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.

ORACLE 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 OpenVMS documentation.
ORA–07643:  smsalo:  SMSVAR is invalid
    Cause  This is an internal error.
    Action  Verify that you can reproduce the error and contact Oracle WorldWide Technical Support and provide the with your INIT.ORA file.

ORA–07647:  sszfck: $OPEN failure
    Cause  While attempting to reopen a file, OpenVMS service $OPEN failed.
    Action  Examine the system message and refer to OpenVMS system documentation.

ORA–07650:  sigtu: $GETJPIW failure
    Cause  While attempting to get the user’s terminal device name during logon, OpenVMS system service $GETJPIW failed.
    Action  Examine the system error message and refer to OpenVMS system documentation.

ORA–07655:  slsprom: $TRNLOG failure
    Cause  While attempting to translate SYS$INPUT during a prompt for a password, OpenVMS system service $TRNLOG failed.
    Action  Examine the system error message and refer to OpenVMS system documentation.

ORA–07656:  slsprom: $GETDVI failure
    Cause  While attempting to get device characteristics during a prompt for a password, OpenVMS system service $GETDVI failed.
    Action  Examine the system error message and refer to OpenVMS system documentation.

ORA–07657:  slsprom: $ASSIGN failure
    Cause  While prompting for a password, OpenVMS system service $ASSIGN failed.
    Action  Examine the system error message and refer to OpenVMS system documentation.
ORA–07658:  

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

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

ORA–07665:  

Cause  
An OpenVMS exception occurred while executing in the Oracle7 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 Worldwide Technical Support.

ORA–07670:  

Cause  
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:  

Cause  
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:  

Cause  
A call to the Oracle7 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 Worldwide Technical Support.

ORA–07681:  

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 Worldwide Technical Support.
ORA-07682: sou2os: set kernel dispatch fail err

Cause  During Oracle7 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 Worldwide Technical Support.

ORA-07683: sou2os: $SETPRV reset error

Cause  During an attempt to restore user privileges at Oracle7 shared image exit, OpenVMS system service $SETPRV failed. This is an internal error.
Action  Verify that you can reproduce the error and contact Oracle Worldwide Technical Support.

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  OpenVMS system service $CREATE_REGION_64 failed.
Action  Examine system error message and refer to OpenVMS system documentation.

ORA-07689: smscre: $CRMPSC_GFILE_64 failure

Cause  OpenVMS system service $CRMPSC_GFILE_64 failed.
Action  Examine system error message and refer to OpenVMS system documentation.

ORA-07690: smscre: $CRMPSC_GDZRO_64 failure

Cause  OpenVMS system service $CRMPSC_GDZRO_64 failed.
Action  Examine system error message and refer to OpenVMS system documentation.

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

Cause  OpenVMS system service: $GRANTID failed.
Action  Add ORA_SGA identifier to the system.

ORA-07692: ssmsget: $MGBSLC_64 failure

Cause  OpenVMS system service $MGBLSC_64 failed.
Action  Examine system error message and refer to OpenVMS system documentation.
ORA–07693: ssmsget: $DELTVA_64 failure
Cause  OpenVMS system service $DELTVA_64 failed.
Action  Examine system error message and refer to OpenVMS system documentation.

ORA–07694: ssmsget: $CREATE_REGION_64 failure
Cause  OpenVMS system service $CREATE_REGION_64 failed.
Action  Examine system error message and refer to OpenVMS system documentation.

ORA–07695: smsfre: $DELETE_BUFOBJ failure
Cause  OpenVMS system service $DELETE_BUFOBJ failed.
Action  Examine system error message and refer to OpenVMS system documentation.

ORA–07696: smsfre: $DELETE_REGION_64 failure
Cause  OpenVMS system service $DELETE_REGION_64 failed.
Action  Examine system error message and refer to OpenVMS system documentation.

ORA–07697: smscre: $GRANTID failure
Cause  OpenVMS system service $GRANTID failed.
Action  Examine system error message and refer to OpenVMS system documentation.

ORA–07698: smsget: $GRANTID failure
Cause  OpenVMS system service $GRANTID failed.
Action  Examine system error message and refer to 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 Worldwide Technical Support.
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 Worldwide Technical Support.

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 Oracle7 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 Oracle7 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 Worldwide Technical Support.

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 Oracle7 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 Oracle7 Server Administrator’s Guide for the proper syntax of the text.
ORA-07708: sksaprs: tape label name buffer too small

**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 Worldwide Technical Support.

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 Worldwide Technical Support.

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**: OpenVMS system service SYS$MOUNT failed.

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

ORA-07715: sksadtd: SYS$DISMNT failure

**Cause**: OpenVMS system service SYS$DISMNT failed.

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

ORA-07716: sksachk: invalid device specification for ARCHIVE

**Cause**: OpenVMS system service SYS$GETDVI failed.

**Action**: Specify a valid device in ARCHIVE control string.
<table>
<thead>
<tr>
<th>ORA–07717:</th>
<th>sksaalo: error allocating memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause</strong></td>
<td>OpenVMS system service LIB$GETVM failed.</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>Check for a system error message and refer to OpenVMS system documentation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORA–07718:</th>
<th>sksafre: error freeing memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause</strong></td>
<td>OpenVMS system service LIB$FREE_VM failed.</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>Check for a system error message and refer to OpenVMS system documentation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORA–07721:</th>
<th>scgcm: not enough OS resource to obtain system enqueue</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause</strong></td>
<td>A call to $ENQ returned an error indicating that the operating system lacked the resources necessary to create a lock. This is caused by the OpenVMS errors SS$_EXENQLM or SS$_INSFMEM.</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>Free up some of the required resource to allow the creation of the required lock.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORA–07740:</th>
<th>slemop: incorrect handle size (programming error)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause</strong></td>
<td>Structures used for reading error message files do not match.</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>This is an internal error. Verify that you can reproduce the error and contact Oracle Worldwide Technical Support.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORA–07741:</th>
<th>slemop: $OPEN failure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause</strong></td>
<td>OpenVMS system service $OPEN failed.</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>Check for a system error message and refer to OpenVMS system documentation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORA–07742:</th>
<th>slemop: $CONNECT failure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause</strong></td>
<td>OpenVMS system service $CONNECT failed.</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>Check for a system error message and refer to OpenVMS system documentation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORA–07743:</th>
<th>slemop: incorrect error file attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause</strong></td>
<td>An error message file is of incorrect format.</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>Unless an error file has been changed, contact Oracle Worldwide Technical Support.</td>
</tr>
</tbody>
</table>
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 Worldwide Technical Support.

ORA-07745: slemcl: $CLOSE failure
  Cause  OpenVMS system service $CLOSE failed.
  Action Check system error and refer to 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 Worldwide Technical Support.

ORA-07747: slemrd: $READ failure
  Cause  OpenVMS system service $READ failed.
  Action Check system error and refer to 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 Worldwide Technical Support.

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 Worldwide Technical Support.

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 Worldwide Technical Support.
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 Worldwide Technical Support.

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 Worldwide Technical Support.

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 Worldwide Technical Support.

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 Worldwide Technical Support.

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 Worldwide Technical Support.

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 Worldwide Technical Support.
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 Worldwide Technical Support.

ORA–07800: slbtpd: invalid number
  Cause An impossible request for binary to decimal conversion was made.
  Action This conversion cannot be performed.

ORA–07801: slbtpd: invalid exponent
  Cause An impossible request for binary to decimal conversion was made.
  Action This conversion cannot be performed.

ORA–07802: slbtpd: 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: SYS$CRELNM failure
  Cause An error was returned from the SYS$CRELNM function.
  Action Check the system error and refer to the 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 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 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 OpenVMS system documentation.

ORA–07824: sspsain: $SETIMR failure
Cause An error was returned from SY$SETIMR while trying to queue a process spin–watch timer.
Action Check the system error message and refer to the OpenVMS system documentation.

ORA–07825: sspsck: #QIO failure at AST level
Cause An error was returned from SY$QIO while trying to read the process dump mailbox.
Action Check the system error message and refer to the OpenVMS system documentation.

ORA–07826: sspscm: SYS$GETDVIW failure
Cause An error was returned from SY$GETDVIW while trying to get information about the process dump mailbox.
Action Check the system error message and refer to the 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 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 OpenVMS system documentation.

ORA-07842:  sllfc1: 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 OpenVMS system documentation.

ORA-07843:  sllfc1: 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 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 OpenVMS system documentation.

ORA-07845:  sllfc1: 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 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 SYSSCONNECT while attempting to open the data file.

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

ORA-07848:  sllfrb:  $GET failure

Cause  An error was returned by SYSSGET while attempting to read the data file.

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

ORA-07849:  sllfsk:  $GET failure

Cause  An error was returned by SYSSGET while attempting to skip records in the input file.

Action  Check the system error message and refer to the 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 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 Worldwide Technical Support.

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 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 OpenVMS system service $OPEN failed.
   Action Examine system error and refer to OpenVMS system documentation.

ORA–07865: ssfduic: $CLOSE failure
   Cause  The OpenVMS system service $CLOSE failed.
   Action Examine system error and refer to OpenVMS system documentation.
This appendix presents some useful tips about managing your 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 Digital 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 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 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 OpenVMS Mailbox or SQL*Net DECnet drivers.) 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 OpenVMS while engaged in a client–server connection to Oracle (connecting to 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.
This appendix is designed to be used as a supplement to the following documents:

- Oracle7 Server Administrator’s Guide
- Oracle7 Application Developer’s Guide
- Oracle7 Server Concepts
- Oracle7 Server Migration Guide
- Oracle7 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 Oracle7 for 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 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 OpenVMS, the Oracle7 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 Oracle7 for OpenVMS.

directing records

Operating system audits are not currently used with Oracle7 for OpenVMS.
A user can be authenticated in many different ways. For example, you can set 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.

For more information about background processes, see Chapter 1, “Introduction to Oracle7 on OpenVMS.”

ARCH For more information about archiving mode, see Archiving Process and Archiving Mode.

DBWR Oracle7 on OpenVMS does not support multiple DBWR processes.

names The prefix ORA_<sid> is concatenated to the beginning of every background process. The <sid> is the value of the logical name ORA_SID. For example, if the value of the logical name ORA_SID is MIS, then the DBWR process is identified as ORA_MIS_DBWR.

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 Oracle7.”

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.”

For more information about communication links, see the SQL*Net documentation.
Clusters, Estimating Size

Use the following figures for calculating cluster size in the Oracle7 Server guides:

- Fixed header size = 86 bytes
- Variable transaction header = 24 (INITRANS value for the table)
- Row header = 4 bytes per row of a clustered table

Configuring Oracle7

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 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 OpenVMS block size, see the Oracle7 for OpenVMS Installation Guide for your platform.

specification

Refer to the paragraph in this section on specifying names for information about filename specification.

Creating a Database During Installation

You must use ORAINST 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
On OpenVMS, the Oracle7 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.

For more information, see “Introduction to Oracle7 on OpenVMS,” in Chapter 1 of this guide.

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
```

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, “Administering your Database and Instances,” and the section on database and tablespace files in the Oracle7 Server Concepts Manual.

Oracle7 for OpenVMS supports 1022 concurrently open files.

No information is currently available.

For the equation given in the Oracle7 Server guides, use the following figures:

- minimum database file size = 5 Mb for the first file.
- database files = 1022 or value of DB_FILES in the INIT.ORA parameter file or value of MAXDATAFILES in CREATE DATABASE.

File size=255

Refer to the SQL*Net Version 2.3.4 for OpenVMS Configuration and User’s Guide for information about database string specification.

Refer to the section on authentication through OS in this appendix for information about user authentication.
Oracle7 for 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 Version 2.3.4 for OpenVMS Configuration and User’s Guide.

**Dispatcher Process**

Refer to the ORA_RDBMS:README.DOC for information on configuring and starting the dispatcher processes of the Multi–Threaded Server (MTS).

**Files**

For more information about designating archive destination devices, see Chapter 6, “Backing Up and Archiving your Database,” of this guide.

ORA_DB:INIT.ORA is the name of the parameter file shared by all instances of a given database.

ORA_DB:<setup_node>_<sid>_INIT.ORA is the name of the parameter file used by each individual instance.

Refer to Chapter 5, “Managing the Database,” in this guide for more information on file names.

**Indexes**

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
SQL> SELECT * FROM v$type_size
    2 WHERE type IN ('KCBH','KTBBH','KTBIT','KDXLE','KDXBR');
```

No information is currently available for these calculations.

**Installing Products**

Refer to the instructions in the Oracle7 for OpenVMS Installation Guide for your platform for complete information about installation.
Instance

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.

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.

Use the STARTUP command.

Latches

No OpenVMS–specific information is currently available.

Listener Process

For information on TNS listener processes, refer to the SQL*Net Version 2.3.4 for 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 500K REUSE

MAXLOGFILES, MAXLOG–MEMBERS

When using ORACLEINS, the 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

There are no OpenVMS–specific elements in the contents of the monitor fields.

No information is currently available on the interrupt mechanism.

No OpenVMS-specific information is currently available.

Multithreaded Server

Refer to the SQL*Net Version 2.3.4 for OpenVMS Configuration and User’s Guide for information about connect strings.
Multiple Instances

Oracle7 for OpenVMS supports multiple instances.

National Language Support (NLS)

error file

The <facility><language>.MSB files in the product directories provide multilingual error messages. Refer to the file ORA_NLS:README.NLS for OpenVMS–specific information on these files.

NLS_LANG

To set the parameter, enter the following command from your system prompt:

$ DEFINE NLS_LANG <language>_<territory>_<charset>

For a list of languages, territories, character sets, and more information, refer to the file ORA_NLS:README.NLS.

For more information about NLS, see Appendix A in this Guide.

Parallel Server

For more information about parallel instances, see the “Managing Instances” chapter in this Guide.

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 Oracle7 for OpenVMS Installation Guide for your platform.
Refer to the ORA_DB:INITPS.ORA file for suggested settings of these parameters.

**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 OpenVMS blocks).
Resource Costs

No information is currently available on setting resource costs.

Roles

For the role specification syntax given in the Oracle7 Server Guides, \(<ID>\) refers 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 Oracle7 for OpenVMS Installation Guide for your platform.

OSOPER and OSDBA

On OpenVMS, the OSOPER role is controlled by the ORAOPER 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 ORAOPER process rights identifier have insufficient privileges to perform the duties authorized by the OSOPER role.

For more information about the ORA_DBA limitation, see the Oracle7 for OpenVMS Installation Guide for your platform.

Script File Names

Script file names are not case dependent in OpenVMS.

Segments Extents

No OpenVMS–specific information is currently available on extents.

SQL Scripts, Names, and Locations

<table>
<thead>
<tr>
<th>Script Name</th>
<th>ORA_RDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATAUDIT.SQL</td>
<td>ORA_RDBMS</td>
</tr>
<tr>
<td>UTLCHAIN.SQL</td>
<td>ORA_RDBMS</td>
</tr>
<tr>
<td>DIT scripts</td>
<td>ORA_RDBMS</td>
</tr>
<tr>
<td>UTLEXCPT.SQL</td>
<td>ORA_RDBMS</td>
</tr>
<tr>
<td>UTLLOCKT.SQL</td>
<td>ORA_RDBMS</td>
</tr>
</tbody>
</table>

These scripts are not provided for OpenVMS.

If you create your database with the CREATE DATABASE command, you must run the DATABASE command, you must run...
CATALOG.SQL, CATEXP.SQL, and ULVIEW.SQL after the CREATE DATABASE command completes.

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 OpenVMS Oracle7 installation scripts.

STANDARD & DBMSSTDX.SQL
DBMSNAP.SQL
ORA_RDBMS

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 V2 listener process. You access the SQL*Net V2 listener process when you use the Multi–Threaded Server configuration of Oracle7.

installing
Refer to the Oracle7 for OpenVMS Installation Guide for your platform for more information on installing SQL*Net.

O/S comm.
software
Refer to the Oracle7 for OpenVMS Installation Guide for your platform for all software and hardware requirements.

specifying links
Refer to the Oracle7 for OpenVMS Installation Guide for your platform and the SQL*Net Version 2.3.4 for OpenVMS Configuration and User's Guide for host string specifications.

System Global Area (SGA)

The SGA is described in Chapter 1, “Introduction to Oracle7 on OpenVMS,” in this guide.

Table Size, Estimating

Clustered
No OpenVMS–specific information is currently available.
The maximum value for MAXEXTENTS is 121.

No OpenVMS-specific information is currently available.

**Tablespaces, Default Storage Parameters**

**Tuning Information**

Refer to Chapter 8, “Optimizing Oracle7,” in this guide.

**Trace Files**

Refer to Chapter 9, “Trace Files,” in this guide.

**Transaction Entries, Space Required Initialization Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>DefaultValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKGROUND_DUMP_DEST</td>
<td>ORA_DUMP</td>
</tr>
<tr>
<td>BG_PRIORITY</td>
<td>1</td>
</tr>
<tr>
<td>COMMIT_POINT_STRENGTH</td>
<td>1</td>
</tr>
<tr>
<td>CPU_COUNT</td>
<td>1</td>
</tr>
<tr>
<td>DB_BLOCK_CHECKSUMS</td>
<td>FALSE</td>
</tr>
<tr>
<td>DB_BLOCK_SIZE</td>
<td>2048</td>
</tr>
<tr>
<td>DB_FILES</td>
<td>32</td>
</tr>
<tr>
<td>DB_FILE_MULTIBLOCK_READ_COUNT</td>
<td>8 (Range of 1–65536/DB_BLOCK_SIZE)</td>
</tr>
<tr>
<td>DB_FILE_SIMULTANEOUS_WRITES</td>
<td>4</td>
</tr>
<tr>
<td>DISTRIBUTED_TRANSACTIONS</td>
<td>16</td>
</tr>
<tr>
<td>INSTANCE_NUMBER</td>
<td>0</td>
</tr>
<tr>
<td>LOG_ARCHIVE_BUFFER_SIZE</td>
<td>127</td>
</tr>
<tr>
<td>LOG_ARCHIVE_BUFFERS</td>
<td>4</td>
</tr>
<tr>
<td>LOG_ARCHIVE_DEST</td>
<td>ORA_ARCHIVE:</td>
</tr>
<tr>
<td>LOG_ARCHIVE_FORMAT</td>
<td>ARCH%T_%S.ARC</td>
</tr>
<tr>
<td>LOG_BLOCKS_DURING_BACKUP</td>
<td>N/A</td>
</tr>
<tr>
<td>LOG_BUFFER</td>
<td>8192(4xDB_BLOCK_SIZE)</td>
</tr>
<tr>
<td>LOG_CHECKPOINT_INTERVAL</td>
<td>10000</td>
</tr>
<tr>
<td>LOG_SMALL_ENTRY_MAX_SIZE</td>
<td>N/A</td>
</tr>
</tbody>
</table>
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 C, “Messages and Codes,” of this guide for a complete list of messages for Oracle7 on OpenVMS.

Installing Export Views
Export views are automatically installed by the Oracle7 installation scripts. If they must be defined at another time, run the script called ORA_RDBMS: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.

MTS_MAX_DISPATCHERS 0
MTS_MAX_SERVERS 0
MTS_SERVERS 0
NLS_LANGUAGE AMERICAN
NLS_TERRITORY AMERICA
OS_AUTHENT_PREFIX ops$
PROCESSES 50
SHARED_POOL_SIZE 350000 bytes
SORT_AREA_SIZE 65536
SORT_READ_FAC 20
SORT_SPACEMAP_SIZE 512
TEMPORARY_TABLE_LOCKS 60
TRANSACTIONS_PER_ROLLBACK_SEGMENT 16
USER_DUMP_DEST ORA_DUMP
Size of SMALLINT Data Type

No information is currently available.

Length Subfield in VARCHAR Data

No information is currently available.

Length Subfield in VARCHAR Data

No information is currently available.

File Processing Options

Current options available for 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 OpenVMS you cannot specify the number of I/O buffers to be used in an operation.
Index

A
Ada, 10 – 7
ALTER DATABASE command, 5 – 7, 6 – 2
ALTER SYSTEM command, 2 – 5
ALTER TABLESPACE command, 5 – 7
ARCH, E – 2
ARCH process, 6 – 4
Architecture, for Oracle7 Parallel Server, 1 – 10
ARCHIVELOG mode, 1 – 7
Archiver process, E – 2
Archiving mode, E – 2
archiving to tape, not supported on OpenVMS, B – 8
Audit trail, E – 2
AUTHORIZE utility, 2 – 3

B
Background processes, E – 3
BACKGROUND_DUMP_DEST parameter, 9 – 2, B – 7
Backup procedures, 1 – 7
Backups, E – 3
Binary integers, 10 – 19
Blocksize, 5 – 7
Buffer cache, 1 – 10
Buffers
  database buffer pool, 1 – 5
  redo log buffers, 1 – 6

C
C Language
  precompiling, 10 – 7
  SQL*Module, 10 – 21
CATDBSYN.SQL script, 7 – 5, 8 – 9
CLLIB.OLB, 10 – 14
Clusters, estimating size, E – 4
COBOL, 10 – 7, 10 – 20
Commands
  ALTER DATABASE, 5 – 7
  CONNECT INTERNAL, C – 2
  CREATE TABLE, 5 – 7
  DROP TABLESPACE, 5 – 8
Compiling with programmatic interfaces, 10 – 9
concurrency, shared mode, 1 – 9
CONNECT INTERNAL, C – 2
CONTINUE command, D – 2
Control files
  creating, 5 – 6
  identifying, 5 – 6
  logical assignment, B – 3
control files, E – 4
Control keys, D – 3
CONTROL–C, D – 2
CONTROL–Y, D – 2
CONTROL_FILES parameter, B – 7
CORELIB.OLB, 10 – 14
CREATE DATABASE
  default value for CONTROLFILE parameter, E – 4
  default value for DATAFILE parameter, E – 4
  default value for LOGFILE parameter, E – 7

D
Data blocks, modified, writing to disk, 1 – 11
Data dictionary creation, E – 4
data files, E – 4
  maximum size of, E – 4
Database
  automatic recovery, 7 – 2
  creation during installation, E – 4
  granting access to, 2 – 3
  limits, E – 4
  opening in exclusive mode, 3 – 4
Database files, defined, 1 – 6
Database links, creating, 5 – 6
Database verification utility, 5 – 11
database verification utility, 5 – 11
DATAFILE parameter, default value, E – 4
Datafiles, 1 – 8
Datatypes, 10 – 19
DB_BLOCK_BUFFERS parameter, 1 – 5
DB_BLOCK_SIZE parameter, B – 7
DB_dbname directory, 5 – 7
DCL command, from within SQL*Plus or SQL*Forms, 11 – 3
DCL–W–ACTIMAGE, error message, C – 1
DEBUG qualifier, 10 – 8
Demo files, programmatic interfaces, 10 – 11
Detached processes, 7 – 3
  identification, B – 5
Disabling control keys, D – 3
Dispatcher process, E – 6
DLM, 1 – 10
DROP TABLESPACE command, 5 – 8

E
EDIT command, 11 – 9
Editing files in SQL*Plus, 11 – 9
error activating image, C – 1
Errors
  ORA–00445, 8 – 3
  ORA–00921, 5 – 7
  ORA–07623, 8 – 3
event flags, restrictions, 10 – 8
Exclusive mode, 3 – 4
exclusive mode, 1 – 8
database access, 1 – 8
EXIT command, D – 2
Export utility, 6 – 7
  for reducing fragmentation, 8 – 8
Export views, installing, E – 13
Exporting, to multiple tapes, 6 – 9

F
file processing options, E – 14
Files, general information, E – 6
FORTRAN, 10 – 7, 10 – 20
Fragmentation, 8 – 8

H
Hardware requirements, and VMScluster configurations, 1 – 11
Help, SQL*Plus Help tables, 11 – 2
HOST
  DCL command, 11 – 3
  option for PRO command, 10 – 7

I
I/O, improving, 1 – 5
I/O buffers, E – 14
Import utility, for recovery, 7 – 5
Importing, from multiple tapes, 6 – 9
INCLUDE option, 10 – 7
index block overhead, E – 6
INIT.ORA file, location, B – 4
INIT.ORA files
at start-up, 1 – 4
setting SGA size, 1 – 4
INSORACLE.COM file, run as a batch job, 3 – 11
Instances
definition, E – 7
failure, 7 – 2
starting in parallel mode, 4 – 3
INSUTILITY.COM, 8 – 8
Interrupt keys
PASTHRU mode, D – 2
SQL*Plus, 11 – 2
Interrupting operations, D – 2

K
KLIB.OLB, 10 – 14
known object limit, 4 – 8
KUSR.OLB, 10 – 14

L
limit, on length of HOST command, 11 – 3
linker directives, with Oracle programming interfaces, 10 – 11
Linking
LNPRO symbol, 10 – 10
LOUTL.COM, 10 – 13
OCI routines, 10 – 16, 10 – 19
LLIB.OLB, 10 – 14
LNOCI.COM, 10 – 18
LNOCIC.COM, 10 – 17
log files, E – 7
LOG_ARCHIVE_DEST parameter, B – 8
LOG_ARCHIVE_FORMAT parameter, B – 8
LOG_ARCHIVE_START parameter, 6 – 3
LOG_BUFFER parameter, 1 – 6
LOG_CHECKPOINT_INTERVAL parameter, 1 – 6, 6 – 2
LOGFILE parameter, default value, E – 7
Logical names
alphabetical listing, B – 3
command files that define, B – 2
Logicals
identification, 1 – 8
using to specify files, 1 – 7
LOGIN.COM file, of ORACLE users, 2 – 2
LOGIN.SQL, 11 – 9
LOGINOUT, D – 3

M
MAXEXTENTS, E – 12
MAXLOGFILES, default value, E – 7
MAXLOGMEMBERS, default, E – 7
Media failure, 7 – 2
mode
database access, 1 – 8
exclusive mode, 1 – 8
MTS, E – 6
MTS_SERVICE, E – 8
Multi–Reel files, exporting, 6 – 8
multi–threaded server, 4 – 8
Multilingual Option. See National Language Support
multiple shared mode, 1 – 9
Multiple tapes
exporting, 6 – 9
importing, 6 – 9

N
National Language Support
See also NLS
7–bit character sets, A – 3
8–bit character sets, A – 4
cient/server environments, A – 2
multi–byte character sets, A – 5
non–supported products, A – 8
overview, A – 2
product translations, A – 8

Index – 3
NLS, A – 1–A – 14, E – 8
See also National Language Support
NLSRTL.OLB, 10 – 14
NOARCHIVELOG mode, 1 – 7

O

OCI routines, 10 – 15, 10 – 19
OCI.OLB, 10 – 14
OpenVMS distributed lock manager, 1 – 10
options file, using for linker directives, 10 – 11
ORA–00445 error, 8 – 3
ORA–00921 error, 5 – 7
ORA–07623 error, 8 – 3
ORA_CONTROL logical, B – 3
ORA_CONTROL1 logical, B – 3
ORA_DATA logical name, 11 – 8, B – 3
ORA_DB logical name, B – 3
ORA_DBA privilege, with STARTUP command, 3 – 5
ORA_DUMP logical name, B – 3
ORA_DUMP directory, 6 – 4
ORA_ERROR logical name, B – 3
ORA_INITSQL logical name, B – 4
ORA_INSTALL logical name, B – 4
ORA_INSTANCE logical name, B – 4
ORA_INSUTL, using to install shared images, 8 – 7
ORA_MAP logical name, B – 4
ORA_OLB logical name, B – 4
ORA_PARAMS logical, 8 – 2
ORA_PARAMS logical name, B – 4
ORA_PATH logical name, 11 – 8, B – 4
ORA_PLS logical name, B – 4
ORA_RDBMS logical name, B – 4
ORA_ROOT directory, B – 5
ORA_SID logical, 3 – 5
default value, 1 – 4
ORA_SID logical name, B – 5
ORA_SLAX logical name, B – 5
ORA_USER.COM file, B – 5
ORA_UTIL logical name, B – 5
Oracle
exclusive mode, 1 – 8
shared mode, 1 – 9
ORACLE call interface, OCI routines, 10 – 15
ORACLE programmatic interfaces
compiling, 10 – 9
datatatypes, 10 – 19
demo files, 10 – 13
FORTRAN %REF directive, 10 – 19
HOST option, 10 – 7
linking, 10 – 10
LNOCI.COM, 10 – 18
LNOCIC.COM, 10 – 17
LNPRO symbol, 10 – 10
OCI routines, 10 – 16, 10 – 19
optional call parameters, 10 – 19
precompiling, 10 – 7
PRO symbol, 10 – 8
Pro*Ada set up, 10 – 20
Oracle Worldwide Technical Support, how to contact, viii
ORACLE.EXE file, defined, 1 – 4
Oracle7 code, 1 – 4
granting access privileges, 2 – 3
linked as shared image, 1 – 4
logical names, B – 2
shutting down, 3 – 13
starting up, 3 – 2
Oracle7 Parallel Server, architecture, 1 – 10
ORACLEINS
to shut down Oracle7, 3 – 14
to start Oracle7, 3 – 3
ORACLEINS.COM file, logical assignments made, B – 2
ORAMBX, 5 – 13
ORAUSER_instance.COM file, 2 – 2
running for ORACLE users, 2 – 2
OS_AUTHENT_PREFIX, E – 3
overhead, of index blocks, E – 6
P
Parallel cache management, 1 – 10
parallel mode. See shared mode
PARALLEL option, 1 – 9
Parallel Query, 4 – 8
Parameter
  LOG_ARCHIVE_DEST, B – 8
  LOG_ARCHIVE_FORMAT, B – 8
  SHARED_POOL_SIZE, B – 9
  SORT_AREA_SIZE, B – 9
Parameter files, general information, E – 8
Parameters
  initialization, E – 12
  optional with programmatic interfaces, 10 – 19
  passing to SQL*Plus, 11 – 4, 11 – 5
Pascal, 10 – 7
password utility, 2 – 3
Performance analysis, 11 – 10
PL/I, 10 – 7
Precompiling
  guidelines and restrictions, 10 – 8
  options, 10 – 8
  syntax and example, 10 – 7
Privileges, and Process Rights Identifier, C – 2
Pro*Ada, 10 – 20
Process Global Area, dump files for, 9 – 2
Process ID, OpenVMS, 9 – 2
Process name, OpenVMS, 9 – 2
Process Rights Identifier, C – 2
PRODUCT_USER_PROFILE, 11 – 2
profile files, SQL*Plus, 11 – 7–11 – 10
Programmatic Interfaces, ORACLE
  programmatic interfaces, 10 – 2

Q
Quotas, process quota logistics, 8 – 3

R
Redo log file, mode of use, 6 – 2
Redo log files
  default names for, 1 – 7
  defining, 1 – 6
  general information, E – 9
  modes of, 1 – 7
  moving, 5 – 10
  reusing, 1 – 7
  size of, 1 – 7
REMUTILITY.COM, 8 – 8
Roles, general information, E – 10
Rolling backward, 7 – 2
Rolling forward, 7 – 2
Runtime libraries, used with Oracle7, 1 – 4

S
Server Manager, 3 – 5, 5 – 2
  to start Oracle7, 3 – 4, 3 – 16
SGA, adjusting, 8 – 4
Shared CORE, 10 – 12
Shared memory; installing products in, 8 – 7
shared mode, database access, 1 – 9
SHARED option, 1 – 9
Shared ORACLE7 image, B – 4
Shared Pool, 1 – 5
Shared SQLLIB, 10 – 12
Shared UPI, 10 – 12
SHARED_POOL_SIZE parameter, B – 9
SHUTDOWN command, 7 – 3
SHUTDOWN file, to stop ORACLE7, 3 – 15
SID
  and instances, 4 – 2
  ORA_SID, B – 5
single shared mode, 1 – 9
SORT_AREA_SIZE parameter, B – 9
SPOOL command, 11 – 9
SQL scripts, E – 10
SQL*Loader, file processing options, E – 14
SQL*Module, 10 – 21
SQL*Net, general information, E – 11
SQL*Plus
  editing files in, 11 – 9
Index – 6

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 – 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
STARTUP file, to start ORACLE7, 3 – 4
System Global Area
See also SGA
components, 1 – 5
defined, 1 – 4
dump files for, 9 – 2
identification, B – 5
size, 1 – 4
SYSTEM tablespace, 7 – 4
removing, 5 – 7

T
Tablespace files, moving, 5 – 8

Terminating operations, D – 2
TIMING command, 11 – 10
Trace files, B – 3
tracing, and SQL trace utility, 9 – 2

U
User authentication, E – 3
User Identification Code (UIC), 9 – 2
USER_DUMP_DEST parameter, B – 9
Utility
database verification, 5 – 11
useful, 5 – 11

V
VLM_BACKING_STORAGE_FILE parameter, B – 10
VMS Mailbox driver, creating database links for the, 5 – 6
VMScluster, 1 – 10
VMScluster environments, 1 – 11
Reader’s Comment Form

Oracle7 for OpenVMS Server and Tools Administrator’s Guide
Part No. A69346–01

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 topic, chapter, and page number below:

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

Please send your comments to:

    Compaq Products Division Documentation Manager
    Oracle Corporation
    500 Oracle Parkway
    Mailstop 659104
    Redwood City, CA  94065
    Phone: (650) 506–7000     Fax: (650) 506–7361

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

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

Thank you for helping us improve our documentation.