Oracle® Database

Administrator's Reference

11*g* Release 2 (11.2) for HP OpenVMS Itanium

E56697-01

June 2015

ORACLE®

Oracle Database Administrator's Reference, 11*g* Release 2 (11.2) for HP OpenVMS Itanium

E56697-01

Primary Author: Bharathi Jayathirtha

Contributors: Dave Hayter, Gary Huffman, Marc Noel, Chris Schuetz, David Miller, Kevin Duffy, Steve Holck, Grant Hayden, Gary Tate

Contents

2 Starting and Stopping Oracle Software

3 Configuring Oracle Products

4 Administering SQL*Plus

5 Configuring Oracle Net Services

6 Using Oracle Precompilers and the Oracle Call Interface

7 Building and Running Demonstrations

C Backing Up and Archiving the Database

D Logical Names and Parameters

E Messages and Codes

F Process Control

Index

Preface

This guide and *Oracle Database Installation Guide for HP OpenVMS Itanium* provide instructions for installing and configuring Oracle Database 11*g* Release 2 (11.2) on HP OpenVMS Itanium systems.

Refer to *Oracle Database Installation Guide for HP OpenVMS Itanium* and to the My Oracle Support Note 377470.1 for the certification matrix and the latest list of certified hardware platforms and operating system versions. You can visit the My Oracle Support website at

https://support.oracle.com/

Audience

This document is intended for anyone responsible for administering Oracle Database 11*g* Release 2 (11.2) on HP OpenVMS Itanium systems.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documentation

Refer to the following documents for information about system administration and tuning for a production database system:

- *Oracle Database Administrator's Guide*
- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Net Services Reference*
- *Oracle Database Performance Tuning Guide*

Refer to *Oracle Database Upgrade Guide* for information about migrating from a previous version of Oracle Database.

The platform-specific documentation for Oracle Database 11*g* products includes the following manuals:

- *Oracle Database Installation Guide for HP OpenVMS Itanium*
- *Oracle Database Release Notes for HP OpenVMS Itanium*

You can check for updates to these documents at

`http://docs.oracle.com/en/database/database.html`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Administering Oracle Database 11*g*

This chapter provides information about administering Oracle Database 11*g* on HP OpenVMS Itanium systems. It contains the following sections:

1.1 Overview

You must set HP OpenVMS logical names, parameters, and user settings for Oracle Database to work. This chapter describes the various settings for Oracle Database on HP OpenVMS.

This section contains the following topics:

1.1.1 About Oracle Database 11*g* Code

Oracle Database 11*g* code consists of several object libraries that form the Oracle Database 11*g* image during installation. The code base consists of more than 70 server and client images, which are linked during installation.

> **See Also:** *Oracle Database Installation Guide for HP OpenVMS Itanium* for more information about shared Oracle Database Client image

Oracle Database 11*g* code is built to use 64-bit pointers to support very large System Global Areas (SGAs). The code for client software is built to use 32-bit pointers to maintain compatibility with the existing client code. Therefore, there are both 32-bit

and 64-bit versions of the installed object and sharable libraries. Oracle only supports 32-bit clients. Client applications may not be built with 64-bit pointers.

1.1.1.1 ORACLE.EXE File

When Oracle Database 11*g* Enterprise Edition is linked and installed, the image resides as sharable code in HP OpenVMS global memory.

You must relink the image when a new code is distributed or when a new release of HP OpenVMS is installed.

1.1.2 Oracle Database 11*g* Instances

An Oracle Database 11*g* instance is a combination of Oracle Database processes and memory buffers.

Because many instances can exist on one system or in one HP OpenVMS Cluster, you must assign a unique, one-to-six character system ID (SID) to every instance. During the installation, you must create an instance when you create the initial database. The SID that you assign to this instance becomes the default value of logical `ORACLE_SID`.

Before you perform any operations on the database instance, you must set the correct Oracle environment and the logicals, for example, `ORA_SID` and `ORACLE_SID`. To do this, run the following command:

```
$ @device:[install_directory]orauser SID
```

This section contains the following topics:

- Section 1.1.2.1, "About System Global Area"
- Section 1.1.2.2, "Storing Data"
- Section 1.1.2.3, "Storing Changes to the Database"
- Section 1.1.2.4, "Using Redo Log Files"
- Section 1.1.2.5, "Using Logical Names"

1.1.2.1 About System Global Area

System Global Area (SGA) is an area of shared memory that is allocated to each Oracle Database instance. All database operations use the data stored in the SGA.

The size of the SGA is determined by parameters in the `INIT.ORA` file. These parameters determine:

- Allocation of Oracle Database resources used by the processes that share the SGA
- Amount of data that may be maintained in the SGA

After you create an instance, you can change the size of its SGA by shutting down the instance with the SQL*Plus utility and modifying the values set in the `INIT.ORA` file as needed.

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

Oracle Database 11*g* Release 2 (11.2) includes support for the Very Large Memory (VLM) 64-bit feature. This allows a large SGA that is limited only by the amount of physical memory available.

Data retrieved or inserted by user transactions is temporarily buffered in the SGA. Because this data resides in an area of memory accessible by all Oracle processes, disk I/O is reduced and transaction time is significantly improved.

The following are the most significant structures in the SGA:

See Also: *Oracle Database Administrator's Guide*

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 multiuser systems. Smaller values use less memory. The limit for this parameter is determined by the size of the SGA. The shared pool must be at least 150 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 the changes are committed and until more buffers are required. If the data is modified, then it is written to the database files.

The number of blocks that can be maintained in the buffer pool is determined by the initialization parameters `DB_CACHE_SIZE` or `DB_nnK_CACHE_SIZE`.

redo log buffer

When the 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 help in data recovery if media or system failure occurs before the modified data is written from the database buffer to the database file.

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

1.1.2.2 Storing Data

Data is stored in database files. When you first create a database, a small number of database files are also created.

When you use Oracle Database Configuration Assistant (Oracle DBCA) to create a database, the database files are stored in the default location, `ORA_ROOT:[ORADATA.`*dbname*`]`, where *dbname* is the database name. You can specify any disk and directory for these database files and this directory need not necessarily be under `ORACLE_HOME`.

The SYSTEM tablespace containing the data dictionary resides in one of the initial database files, which may also contain data entered by Oracle users. You can expand the database by creating tablespaces and database files.

Oracle recommends that the cluster size on the disk drives that contains the database files be an integer multiple of the Oracle block size. For example, if the blocks are 8 KB, which is the default, then the cluster size must be 8 KB, 16 KB, 24 KB, and so on. Cluster sizes are specified in disk blocks, where one block is equal to 512 bytes. Therefore, for an 8 KB Oracle block, the recommended cluster sizes in disk blocks can be 16, 32, 48, and so on.

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

1.1.2.3 Storing Changes to the Database

Changes made to the database are logged in the database buffer pool and in the redo log file. The changes recorded in the redo log file helps in data recovery when 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 is available when the current log is filled.

Modified data is written from the database buffer pool to the database files when the current redo log is full or when the number of blocks in the redo log equals the value set by the INIT.ORA parameter LOG_CHECKPOINT_INTERVAL. Any event that causes the database buffers to be written is known as a checkpoint.

The default value of the LOG_CHECKPOINT_INTERVAL parameter is 10,000 disk blocks, which is equal to 5 MB. If required, you can specify a different value. The values assigned to this parameter must be in multiples of physical block size. By setting the LOG_CHECKPOINT_INTERVAL initialization parameter to zero, you can eliminate interval checkpoints. This reduces the checkpoint frequency and optimizes run-time performance.

1.1.2.4 Using Redo Log Files

There are 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, or system failure.

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

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

In the NOARCHIVELOG mode, data in the log file is overwritten when a redo log file is reused. However, data is not overwritten until data in the database buffer has been written to the database file. Using the redo log files in the NOARCHIVELOG mode ensures data recovery only for software and system failure.

During Oracle Database installation, redo log files are created with the default names REDO01.LOG, REDO02.LOG and REDO03.LOG. By default, these files are saved in the ORA_DB directory, but you can choose an alternative directory. These log files are used in NOARCHIVELOG mode by default. You can change the mode to ARCHIVELOG. By default, these files are 100 MB each. You can alter this size and specify different file names during the installation.

1.1.2.5 Using Logical Names

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

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

Caution: *Never* use process-level concealed logical names to name any Oracle Database, redo log, or control file. Read the information given in *Oracle Database Administrator's Guide* before renaming these files.

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

1.1.3 Logical Names Data Files: Locations and Identifying

Oracle data files may be placed in any location on any disk, subject to the following restrictions:

- The data files or the directory that contains the data files cannot be owned by anyone with a group equal to or less than `MAXSYSGROUP`.
- The Oracle Database 11*g* account must have write access to the location of the data files.
- Data files should not reside in the root level directory of a disk.

You can identify data files by logical names rather than fully qualified file names in the `CREATE DATABASE` or `ALTER TABLESPACE` statements. However, these logical names must be defined at the `GROUP` level or higher, preferably at the `SYSTEM` level. Logical names at the `PROCESS` or `JOB` level cannot be used to identify data files. If you identify the data files by logical names, then ensure that these logical names are defined during system startup before calling any procedure to restart the database.

1.2 Initialization Parameter

The `DB_BLOCK_SIZE` initialization parameter specifies the standard block size for the database. This block size is used for the `SYSTEM` tablespace and by default in other tablespaces.

You can set the `DB_BLOCK_SIZE` parameter value to a maximum of 32 KB.

Note: You cannot change the value of the `DB_BLOCK_SIZE` parameter after you create a database.

1.3 Database Limits

Table A–1 provides information about the default and maximum values for parameters in a `CREATE DATABASE` or `CREATE CONTROLFILE` statement.

Note: Interdependencies among these parameters may affect allowable values.

Table 1–1 lists the Oracle Database 11*g* file size limits specific to HP OpenVMS.

Table 1–1 File Size Limits

File Type	Maximum Size
Data File	4,194,303 multiplied by the value of the DB_BLOCK_SIZE parameter
Import file	Unlimited (limited by the operating system file size limit)
Export file	Unlimited (limited by the operating system file size limit)
SQL*Loader file	Unlimited (limited by the operating system file size limit)

1.4 Operating System Accounts and Groups

Special operating system accounts and groups are required by Oracle Database 11*g*.

Table 1–2 describes the `oracle` and `system` operating system accounts.

Table 1–2 HP OpenVMS Accounts

Name	Description
`oracle`	This account, called the Oracle software owner, is the operating system account that owns the Oracle Database 11*g* software. The remainder of this document refers to this account simply as the `oracle` user.
`system`	The `system user` is a special operating system account with maximum privileges. This account is used to configure and install networking software, and create user accounts. This user is not to be confused with the Oracle account SYSTEM.
`ORA_NOBODY`	This account, which echoes the UNIX 'nobody' account, is the operating system account through which 'external jobs' are executed. Section 1.5.6, "Security for External Jobs" of this document discusses this account in detail.

1.5 Security

Oracle Database 11*g* uses several features of the HP OpenVMS operating system to provide a secure environment for users. These features include file ownership, group accounts, and the ability of a program to change its user ID when it is run.

The two-task architecture of Oracle Database 11*g* improves security by dividing work (and address space) between the user program and the `oracle` server process. All database access is achieved using the shadow process and special authorizations in the `oracle` server process.

This section covers the following topics:

- Section 1.5.1, "Security for SQL*Plus Command"
- Section 1.5.2, "Security for Database Files"
- Section 1.5.3, "Customizing the Initialization File"
- Section 1.5.4, "Running the orapwd Utility"
- Section 1.5.5, "Password Management"
- Section 1.5.6, "Security for External Jobs"

> **See Also:** *Oracle Database Administrator's Guide* for more information about security issues

1.5.1 Security for SQL*Plus Command

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

1.5.2 Security for Database Files

The `oracle` user must own the database files. Set the read and write permissions on these files for the owner. Do not grant permission for System, Group, and World.

The `oracle` user must also own the directories containing the database files. For additional security, you can curtail visibility of the datafiles. To do this, remove any directory permission for System, Group, and World.

1.5.3 Customizing the Initialization File

The default initialization file, `init.ora` is provided with the Oracle Database 11*g* software. All Oracle Database 11*g* instances assume these values if you do not specify different values for them in the `init`*`sid`*`.ora` file. Oracle recommends that you include only those parameters in the `init`*`sid`*`.ora` file that differ from the default initialization parameter values.

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

> **See Also:** *Oracle Database Administrator's Guide* and *Oracle Database Performance Tuning Guide*

Caution: Ensure that your database uses the following values for the listed parameters:

- `disk_asynch_io=false`
- `tape_asynch_io=false`
- `backup_tape_io_slaves=false`
- `db_Writer_processes=1`

By default, Database Configuration Assistant creates the two asynch parameters with true values.

Table 1–3 lists default initialization parameter values on HP OpenVMS and their default values and range of values.

Table 1–3 Initialization Parameters

Parameters	Default Value	Range of Values
BITMAP_MERGE_AREA_SIZE	1048576	From 65536 onward (no upper limit)
COMMIT_POINT_STRENGTH	1	0 to 255
CONTROL_FILES	none	none
CREATE_BITMAP_AREA_SIZE	8388608	From 65536 onward (no upper limit)
DB_nnK_CACHE_SIZE	0	2 KB to 32 KB
DB_BLOCK_SIZE	8192	2 KB to 32 KB

Table 1–3 (Cont.) Initialization Parameters

Parameters	Default Value	Range of Values
DB_CACHE_SIZE	0	From 8 MB onward (no upper limit)
DB_FILES	200	1 to 2000000
DB_FILE_MULTIBLOCK_READ_COUNT	16	1 to the lower of the following: ■ The value of the DB_nnK_CACHE_SIZE parameter divided by 4 ■ 1048576 divided by the value of the DB_BLOCK_SIZE parameter
DISTRIBUTED_TRANSACTIONS	1/4 TRANSACTIONS	From 0 onward (no upper limit)
HASH_AREA_SIZE	2*SORT_AREA_SIZE	From 0 onward (no upper limit)
JAVA_POOL_SIZE	210 MB	Between 1000000 and 1000000000
LOG_BUFFER	512 KB or 128 KB multiplied by the value of the CPU_COUNT parameter, which ever is higher	66560 to unlimited
LOG_CHECKPOINT_INTERVAL	0	0 to unlimited
DISPATCHERS	5	Between MAX_DISPATCHERS and PROCESSES
MAX_SHARED_SERVERS	2 multiplied by the value of the SHARED_SERVERS parameter, if the value of the SHARED_SERVERS parameter is greater than 20; otherwise, 20	Between SHARED_SERVERS and PROCESSES
SHARED_SERVERS	1, if DISPATCHERS is specified, else 0	Between 1 and PROCESSES
NLS_LANGUAGE	AMERICAN	Valid language names
NLS_TERRITORY	AMERICA	Valid territory names
OBJECT_CACHE_MAX_SIZE_PERCENT	10	0 to unlimited
OBJECT_CACHE_OPTIMAL_SIZE	100 KB	10 KB to unlimited
OPEN_CURSORS	300	1 to unlimited
OS_AUTHENT_PREFIX	OPS$	Arbitrary string

See Also: *Oracle Database Release Notes for HP OpenVMS Itanium* for a complete list of desupported initialization parameters

1.5.4 Running the orapwd Utility

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

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

```
$ ORAPWD FILE=filename PASSWORD=password ENTRIES=max_users
```

Table 1–4 describes the *filename*, *password*, and *max_users* syntax for running the orapwd utility.

Table 1–4 Syntax for Running the orapwd Utility

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

See Also: *Oracle Database Administrator's Guide* for more information about using the `orapwd` utility

1.5.5 Password Management

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

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

1.5.6 Security for External Jobs

To securely execute external jobs, for example using `DBMS_SCHEDULER.CREATE_JOB`, it is necessary to create a minimally privileged HP OpenVMS account called `ORA_NOBODY`. The external jobs will run under this account. The account should be created by the HP OpenVMS System Administrator. For example:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN> SHOW MAXSYSGROUP
```

Make a note of the current value, for example, 8.

```
SYSGEN> EXIT
$ RUN AUTHORIZE
UAF> LIST * /BRIEF
%UAF-I-LSTMSG1, writing listing file
%UAF-I-LSTMSG2, listing file SYSUAF.LIS complete
UAF> EXIT
$ SORT/KEY=(POSITION=35,SIZE=12) SYSUAF.LIS SYSUAF.TMP
$ EDIT/READ SYSUAF.TMP
```

Determine a suitable new `UIC` group which can be used for the new `ORA_NOBODY` account. There should currently be no other accounts in this group and the group must be greater than `MAXSYSGROUP`. For example:

```
Greg Mayhew          GMAYHEW         [240,22]     24959    Normal
Gordon Brown         GBROWN          [240,23]     24959    Normal
Jeffrey Archer       JARCHER         [240,24]     24959    Normal
Charles Windsor      CWINDSOR        [240,25]     24959    Normal
Oracle DBA1          DBA1            [244,1]      27335    All
Oracle DBA2          DBA2            [244,2]      27335    All
Oracle Admin         ORA_ADMIN       [244,3]      27335    Devour
```

In the example, it could be determined that the UIC group 241 could be used

Exit from the editor.

```
$ DELETE SYSUAF.LIS;0
$ DELETE SYSUAF.TMP;0
$ RUN AUTHORIZE
UAF> ADD ORA_NOBODY /ACCOUNT="Nobody"/UIC=[241,1]/PASSWORD=PRIVATE1 -
     /NOPWDEXP/PGFLQUOTA=200000/BYTLM=4000000/BIOLM=100/DIOLM=100 -
     /ASTLM=100/ENQLM=200/TQELM=100/FILLM=100/PRCLM=10/JTQUOTA=512000 -
     /WSDEF=512000/WSQUO=512000/WSEXTENT=1024000/PRIV=(NETMBX,TMPMBX) -
     /DEFPRIV=(NETMBX,TMPMBX)/DEVICE=DKA100:/DIR=[ORA_NOBODY] -
     /FLAGS=(DISCTLY,DISMAIL)
```

Note: The values are just examples.

```
UAF> EXIT
$ CREATE/DIR/LOG/OWNER=ORA_NOBODY DKA100:[ORA_NOBODY]      !  Example
```

1.6 Using Trace and Alert Files

This section describes the trace (or dump) and alert files that Oracle Database 11*g* creates to diagnose and resolve operating problems, and includes:

- Section 1.6.1, "Trace Files"
- Section 1.6.2, "Alert Files"

1.6.1 Trace Files

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

sid_processname_processid.trc,

In this file name format:

- *sid* is the instance system identifier.
- *processname* is the process that generates it.
- *processid* is the HP OpenVMS process id displayed when a HP OpenVMS show system command is entered at DCL.

A sample trace file name is:

```
ORA_ROOT:[000000.diag.rdbms.sri3.sri3.trace]sri3_dbrm_214670ac.trc
```

1.6.2 Alert Files

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

1.7 PL/SQL Gateway

The mod_plsql module is a PL/SQL gateway running within an Apache module in the middle tier server. It runs PL/SQL procedures in a back-end Oracle Database using OCI. The mod_plsql module currently supports only stateless PL/SQL Web applications.

> **See Also:** *Oracle HTTP Server mod_plsql User's Guide* for information about developing Web applications using PL/SQL

The PL/SQL Gateway is installed as part of the Apache Install on HP OpenVMS.

> **See Also:** The `README_MODPLSQL.TXT` file in the Apache directory

1.8 Oracle HTTP Server Powered by Apache

Refer to *Oracle Database Installation Guide for HP OpenVMS Itanium* for information about installing and configuring Oracle HTTP Server powered by Apache.

1.9 Granting Access to Oracle Database 11*g* Users

To grant users access to Oracle Database 11*g*:

> **Note:** The `ORAUSER.COM` script must be located in the top-level Oracle home directory. Do not move this script. The definitions of Oracle logicals are created from the top-level Oracle home directory. If you run the `ORAUSER.COM` script from any other location, then Oracle Database 11*g* will not work correctly.

1. You can define a symbol in the systemwide login procedure, typically `SYLOGIN.COM` that runs a particular `ORAUSER.COM` file. This method may be more useful if users access multiple instances and, therefore, need to run a database-specific `ORAUSER` file with the proper parameters. For example:

   ```
   $ go_prod:== @DISK$DISK1:[ORACLE11G]ORAUSER PROD
   ```

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

   > **See Also:** *Oracle Database Installation Guide for HP OpenVMS Itanium* for more information about account quotas

3. Create the Oracle Database 11*g* user accounts with the `CREATE USER` and `ALTER USER` commands. Use the `GRANT` command to grant the required database privileges or roles as documented in *Oracle Database Administrator's Guide*.
4. To enable users to use the SQL*PLUS utility to start or shut down an Oracle Database 11*g* instance, use the HP OpenVMS `AUTHORIZE` utility to add an `ORA_`*`sid`*`_DBA` or `ORA_DBA` process rights identifier to the user's HP OpenVMS account from the HP OpenVMS rights database.

   > **See Also:** *Oracle Database Installation Guide for HP OpenVMS Itanium*

2

Starting and Stopping Oracle Software

This chapter describes different ways to start or shut down Oracle Database 11*g*. It contains the following topics:

- Section 2.1, "Starting Up Oracle Database 11g"
- Section 2.2, "Shutting Down Oracle Database 11g"
- Section 2.3, "Oracle Net Listener"

2.1 Starting Up Oracle Database 11*g*

Before you start Oracle Database 11*g*, ensure that both an instance and a database exist on the local system. If you did not install Oracle Database 11*g*, then consult the database administrator (DBA).

This section contains the following topics:

- Section 2.1.1, "Before Starting Up"
- Section 2.1.2, "Starting Oracle Database 11g by Using SQL*Plus"
- Section 2.1.3, "Starting Oracle Database 11g Remotely by Using SQL*Plus from an HP OpenVMS Client"
- Section 2.1.4, "Starting Oracle Database 11g Remotely by Using SQL*Plus from a Microsoft Windows PC Client"

2.1.1 Before Starting Up

Note: If you restarted the HP OpenVMS system, for example, due to a system failure, then you should read this section.

After restarting HP OpenVMS and before starting Oracle Database 11*g*, you must run the `ORAUSER.COM` file. In this command, you must specify the full directory path. For example:

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

When the DBA runs this file and the logical name `ORA_AUTO_INSORACLE` is defined as TRUE, a check is performed to determine if `INSORACLE` must be run. If required, `INSORACLE` is run. `INSORACLE` installs the shared global sections that make a sharable `ORACLE` image known to the system.

The following images are installed:

- libclntsh.so
- libskgxp11.so
- libskgxn2.so
- liborashr11.so
- libcorenls11.so
- oracle.exe
- ora_java_vms_shr.exe

Note: By default, ORA_AUTO_INSORACLE is not defined as TRUE and ORAUSER will not attempt to run INSORACLE.

2.1.2 Starting Oracle Database 11*g* by Using SQL*Plus

You can start an instance of Oracle Database 11*g* using SQL*Plus. Refer to the instructions in this document on setting up SQL*Plus. Refer to the generic (platform-independent) Oracle Database documentation for instructions on using SQL*Plus.

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

2.1.2.1 Identifying the Current Instance

When starting up Oracle Database 11*g*, you start the current instance. The current Oracle Database 11*g* instance is identified by the value of the logical name ORACLE_SID. For example, if the value of ORACLE_SID is currently V9, then the current instance is the instance with the SID V9. If you have not reassigned the ORACLE_SID logical name, then the value of ORACLE_SID is the *SID* specified during installation. To change the current instance before starting Oracle Database 11*g* with SQL*Plus, you should run the ORAUSER.COM file with the appropriate *SID* as parameter.

If ORACLE_SID is undefined or incorrect, then the following error message is displayed:

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

2.1.2.2 Specifying Startup Parameters

When the current Oracle Database 11*g* instance is started, the SGA is created and initialized with the startup parameters set in the distributed parameter file, INIT.ORA, in the ORA_DB directory. When using SQL*Plus, you can use another startup file that sets different parameter values by including the PFILE option with the STARTUP command to identify an alternative parameter file. If the file is not in the current default directory, then you must include the directory location of the file:

```
SQL> STARTUP PFILE=INITsid.ORA
```

2.1.2.3 Starting the Server Using SQL*Plus

To start Oracle Database 11*g*, you must have the process rights identifier ORA_DBA or ORA_*sid*_DBA assigned to the user account in the HP OpenVMS rights database. In addition, you must run the COM file that makes the logical name assignments required to run Oracle Database 11*g*.

Before starting up Oracle Database 11*g*, run the ORAUSER.COM file to set the instance.

After running the ORAUSER.COM file, run SQL*Plus and run the appropriate STARTUP commands, as documented in *Oracle Database Administrator's Guide*. You can run the single SQL*Plus command, STARTUP, or run the three separate SQL*Plus commands documented in *Oracle Database Administrator's Guide* to start Oracle Database 11*g* Enterprise Edition.

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

2.1.3 Starting Oracle Database 11*g* Remotely by Using SQL*Plus from an HP OpenVMS Client

You can use SQL*Plus on an HP OpenVMS client to start an Oracle Database 11*g* database instance on a remote HP OpenVMS system.

2.1.3.1 Steps to Perform on a Remote System Database

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

1. Create a password file using ORAPWD. The password file can be either exclusive or shared. For this example, assume that an exclusive password file is used. 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:

   For an exclusive password file:

   ```
   $ DEFINE/SYSTEM/EXEC ORA_sid_PWFILE -
     @DISK:[directory]fname
   ```

   For a shared password file:

   ```
   $ DEFINE/SYSTEM/EXEC ORA_PWFILE -
     @DISK:[directory]fname
   ```

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

   For an exclusive password file:

   ```
   REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE
   ```

   For a shared password file:

   ```
   REMOTE_LOGIN_PASSWORDFILE = SHARED
   ```

4. Stop and restart the database instance.
5. Copy INIT*sid*.ORA from the server to any directory on the client.
6. Start the SQLNET listener on the local system. The SQLNET listener must be configured to service connections specified by the TNSNAMES.ORA entry, which is defined in the following section.

2.1.3.2 Steps to Perform on a HP OpenVMS Client System

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

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

```
$ DEFINE ORA_DFLT_HOSTSTR Oracle_Net_Alias
```

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

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

4. Start SQL*PLUS, and run the commands as follows. When prompted for the password, enter the password you specified in Step 1 of the preceding section when the password file was created.

```
$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 - Production on Sat May 23 05:26:12 2015

Copyright (c) 1982, 2015, Oracle.  All Rights Reserved.
Connected to an idle instance.
SQL> startup  pfile=<pfile location>
ORACLE instance started.

Total System Global Area  1603887104 bytes
Fixed Size                  2213848 bytes
Variable Size             469768232 bytes
Database Buffers         1124073472 bytes
Redo Buffers                7831552 bytes
Database mounted.
Database opened.
SQL>
```

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

2.1.4 Starting Oracle Database 11*g* Remotely by Using SQL*Plus from a Microsoft Windows PC Client

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

2.1.4.1 Steps to Perform on Remote System Database

Perform the steps listed under Section 2.1.3.1, "Steps to Perform on a Remote System Database".

2.1.4.2 Steps to Perform on a Microsoft Windows Client System

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

1. Ensure that there is a TNSNAMES.ORA entry for the *SID* on the remote system where the database resides.
2. Start SQL*Plus on your Microsoft Windows Client System.

   When prompted for the password, enter the password specified in Step 1 of the preceding procedure when the password file was created. Assume the TNSNAMES.ORA alias for the remote database is *net_V11_alias*.

```
SQL> connect @net_V11_alias
Password:
SQL> startup pfile=<parameter_file>
ORACLE instance started.
Total System Global Area              555189984 bytes
Fixed Size                               730848 bytes
Variable Size                         520093696 bytes
Database Buffers                       33554432 bytes
Redo Buffers                             811008 bytes
Database mounted.
Database opened.
SQL> exit
```

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

2.2 Shutting Down Oracle Database 11*g*

This section describes the following methods for shutting down Oracle Database 11*g*:

- Section 2.2.1, "Shutting Down Oracle Database 11g Using SQL*Plus"
- Section 2.2.2, "Stopping Oracle User Processes Before Database Shutdown"
- Section 2.2.3, "Removing Sharable Images"

After all instances on a node have been shut down, you can remove the sharable images.

2.2.1 Shutting Down Oracle Database 11*g* Using SQL*Plus

You can shut down an instance of Oracle Database 11*g* by using SQL*Plus. Refer to Chapter 4, "Administering SQL*Plus" for instructions on setting up SQL*Plus. Then, refer to the generic (platform-independent) Oracle Database documentation for instructions on using SQL*Plus.

2.2.2 Stopping Oracle User Processes Before Database Shutdown

The `SHUTDOWN IMMEDIATE` command may not work if you have persisting connections in the environment. For example, if you run processes associated with monitoring tools such as Oracle Enterprise Manager Agent. In that case, it is advisable to terminate the connections from the operating system level by running the following command:

```
$ STOP/ID=pid
```

Use the procedure below to identify which processes can be stopped.

For example, from a separate SQL*Plus session, run the following query:

```
SQL> SELECT sid,serial#,process FROM v$session WHERE type!='background'AND
program!='Oracle';
```

Suppose the processes that are listed in the following table are the ones that are currently running.

SID	Serial	Process
1	1	20C0018B
2	1	20C0018C

SID	Serial	Process
3	1	20C0018D
4	1	20C0018E
5	1	20C0018F
6	1	20C002DD

Then, you can run commands similar to the following to stop these processes:

```
SQL>HOST STOP/ID=20C0018B
SQL>HOST STOP/ID=20C0018C,
```

2.2.3 Removing Sharable Images

After shutting down all Oracle Database 11*g* instances on a node, to perform code relink, remove the sharable images by running the following command:

```
$ REMORACLE
```

2.3 Oracle Net Listener

This section describes how to stop and start Oracle Net listener.

Stopping Oracle Net Listener

To stop an Oracle Net listener:

1. Set up the Oracle environment for the Oracle home by running `ORAUSER.COM` with the required *`sid`* parameter.

   Note: Do not relocate `ORAUSER.COM` during the installation. It must reside at the top level of the Oracle installation directory.

2. Stop the Oracle Net listener by running the following command:

   ```
   $ LSNRCTL STOP listenername
   ```

   Note: If the name of the listener is the default name `LISTENER`, then you do not have to specify the name in this command.

Restarting Oracle Net Listener

To start an Oracle Net listener:

1. Set up the Oracle environment for the Oracle home by running `ORAUSER.COM` with the appropriate *`sid`* parameter.

   Note: Do not relocate `ORAUSER.COM` during the installation. It must reside at the top level of the Oracle installation directory.

2. Restart the Oracle Net listener using the following command:

   ```
   $ LSNRCTL START listenername
   ```

Note: If the name of the listener is the default name `LISTENER`, then you do not have to specify the name in this command.

2.3.1 Oracle Management Agent

If you are using Oracle Enterprise Manager Grid Control to manage multiple Oracle products from a central location, then you must have an Oracle Management Agent installed on each host system. Typically, an Oracle Management Agent is installed in its own Oracle home directory.

This section describes how to stop and start an Oracle Management Agent.

Stopping Oracle Management Agent

To stop an Oracle Management Agent:

1. Set up the Oracle environment for the Oracle home by running `ORAUSER.COM` with the required *`sid`* parameter.

   Note: Do not relocate `ORAUSER.COM` during the installation. It must reside at the top level of the Oracle installation directory.

2. Enter the following command:

   ```
   $ EMCTL STOP AGENT
   ```

Starting Oracle Management Agent

To start Oracle Management Agent:

1. Set up the Oracle environment for the Oracle home by running `ORAUSER.COM` with the appropriate *`sid`* parameter.

   Note: Do not relocate `ORAUSER.COM` during the installation. It must reside at the top level of the Oracle installation directory.

2. Enter the following command:

   ```
   $ EMCTL START AGENT
   ```

3

Configuring Oracle Products

This chapter describes how to configure Oracle Database and Oracle software on HP OpenVMS. It contains the following sections:

- Section 3.1, "Configuring the Database for Additional Oracle Products"
- Section 3.2, "Using Configuration Assistants as Standalone Tools"
- Section 3.3, "Relinking Executables"

3.1 Configuring the Database for Additional Oracle Products

If you install additional Oracle products after the initial installation, then use Oracle Database Configuration Assistant to configure the database for the new products, as follows.

1. Start the database, if necessary.
2. Before you start Oracle Database Configuration Assistant, set up the VNCSERVER and enter the following commands on HP OpenVMS:

```
$ SET DISPLAY/CREATE/TRANSPORT=TCPIP/NODE=XXXXXX/SERVER=n

$ dbca
```

3. Select **Configure Database Options**.
4. From the list of available databases, select the database that you want to configure.
5. Choose the products that you want to enable from the list, and then click **Finish**.

3.2 Using Configuration Assistants as Standalone Tools

Configuration Assistants are usually run during an installation session, but you can also run them in standalone mode. As with Oracle Universal Installer, you can run each of the assistants noninteractively using a response file. This section contains the following topics:

- Section 3.2.1, "Using Oracle Net Configuration Assistant"
- Section 3.2.2, "Using Oracle Database Configuration Assistant"
- Section 3.2.3, "Configuring New or Upgraded Databases"

3.2.1 Using Oracle Net Configuration Assistant

When Oracle Net Server or Oracle Net Client is installed, Oracle Universal Installer automatically launches Oracle Net Configuration Assistant.

If you choose to perform a separate Oracle Database Client installation, then Oracle Net Configuration Assistant automatically creates a configuration that is consistent with the selections made during the installation. Oracle Universal Installer automatically runs Oracle Net Configuration Assistant to set up a net service name in the local naming file located in the `ORA_ROOT:[network.admin]` directory of the client installation.

Before you start Oracle Net Configuration Assistant, set up the VNCSERVER and then enter the following commands on HP OpenVMS:

```
$ SET DISPLAY/CREATE/TRANSPORT=TCPIP/NODE=XXXXXX/SERVER=n

$ netca
```

Note: When you use Oracle Database Configuration Assistant to create a database, this assistant automatically updates the network configuration files to include information for the new database.

3.2.2 Using Oracle Database Configuration Assistant

You can use Oracle Database Configuration Assistant to:

- Create a default or customized database
- Configure an existing database to use Oracle products
- Generate a set of DCL and SQL scripts that you can inspect, modify, and run at a later time to create a database

Before you start Oracle Database Configuration Assistant, set up the VNCSERVER and then enter the following commands on HP OpenVMS:

```
$ SET DISPLAY/CREATE/TRANSPORT=TCPIP/NODE=XXXXXX/SERVER=n

$ dbca
```

For information about the command line options available with Oracle Database Configuration Assistant, use the `-help` or `-h` command line arguments, as follows:

```
$ dbca -help
```

3.2.3 Configuring New or Upgraded Databases

Oracle recommends that you run the `utlrp.sql` script after creating or upgrading a database. This script recompiles all PL/SQL modules that may be in an invalid state, including packages, procedures, and types. Although it is optional, Oracle recommends that you perform this step when you create the database and not later.

To run the `utlrp.sql` script, follow these steps:

1. Log in as the `oracle` user.
2. Set up the Oracle environment for the Oracle home by running `ORAUSER.COM` with the required *`sid`* parameter.
3. Start SQL*Plus, as follows:

   ```
   $ sqlplus / as sysdba
   ```

4. If necessary, start the database:

```
SQL> STARTUP
```

5. Run the `utlrp.sql` script:

```
SQL> @ORA_ROOT:[RDBMS.ADMIN]UTLRP.SQL
```

3.3 Relinking Executables

Product executables can be re-linked using the `ORA_ROOT:[BIN]RELINK.COM` command procedure. It is necessary to relink after an operating system upgrade and after most Oracle software patches. Some operating system patches may also require relinking of Oracle products. Firstly, shut down the database (SHUTDOWN NORMAL), any listeners, and other processes linked to this installation. Then, use the following commands to complete the relink:

```
$ REMORACLE
$ @ORA_ROOT:[BIN]RELINK ALL
$ @ORA_ROOT:[BIN]RELINK DEPLOY
$ INSORACLE
```

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

3.3.1 Support for Shared Libraries in Recovery Manager

Starting with Oracle9*i* Release 2 (9.2.0.2), support for shared libraries in Recovery Manager (RMAN) has been available. If you want to change the specified Media Management Library (MML), then you need not perform a relink. You can explicitly override the specified MML by using the `ALLOCATE CHANNEL` command as follows:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt PARMS='SBT_LIBRARY=mymml';
}
```

In this example, *mymml* is the full path for the shared library that is provided by the third-party vendor.

RMAN on HP OpenVMS defaults to the Serial Backup Tape (SBT) disk implementation (which is statically linked), unless it is explicitly overridden with the `SBT_LIBRARY` parameter on the `ALLOCATE CHANNEL` command.

Since the shared library is called into the server, third-party vendors who are supplying their own MML must ensure that their library is compiled 64-bit. The Oracle Database 10*g* code was compiled by using the qualifier `/NAMES=AS_IS`, and the symbol vector must exactly match the entry point names as indicated in the SBT API specification. The compilation qualifiers `/FLOAT=IEEE/IEEE_MODE=DENORM/GRANULARITY=BYTE` should also be used.

4

Administering SQL*Plus

This chapter describes how to use and administer SQL*Plus on HP OpenVMS. It contains the following topics:

4.1 Administering Command-Line SQL*Plus

This section describes how to administer command-line SQL*Plus. It includes:

See Also: *SQL*Plus User's Guide and Reference*

4.1.1 Using Setup Files

When you start SQL*Plus, it runs the `GLOGIN.SQL` site profile setup file and then runs the `LOGIN.SQL` user profile setup file.

4.1.1.1 Using the Site Profile File

The global site profile file is `ORA_ROOT:[SQLPLUS.ADMIN]GLOGIN.SQL`. If a site profile already exists at this location, then it is overwritten when you install SQL*Plus. If SQL*Plus is removed, then the site profile file is also removed.

4.1.1.2 Using the User Profile File

The user profile file is `LOGIN.SQL`. SQL*Plus looks for this file in the current directory, and then in the directories you specify by using the `SQLPATH` logical. Set this logical to a comma-delimited list of directories. For example:

```
$DEFINE SQLPATH "disk1:[path1],disk2:[path2]"
```

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

The options set in the `LOGIN.SQL` file override those set in the `GLOGIN.SQL` file.

See Also: *SQL*Plus User's Guide and Reference* for more information about profile files

4.1.2 Using the PRODUCT_USER_PROFILE Table

During a typical installation, the PRODUCT_USER_PROFILE table is created automatically. This table is used to disable the SQL and SQL*Plus commands you specify. To re-create this table, run the following script in the SYSTEM schema:

```
ORA_ROOT:[SQLPLUS.ADMIN]PUPBLD.SQL
```

For example:

```
$ SQLPLUS SYSTEM/MANAGER
SQL> @ORA_ROOT:[SQLPLUS.ADMIN]PUPBLD.SQL
```

4.1.3 Using Demonstration Tables

Oracle Database provides demonstration tables that you can use for testing. To install the demonstration tables in a database, you must choose an installation type that installs a preconfigured database.

See Also: *Oracle Database Installation Guide for HP OpenVMS Itanium* for more information about installation options

4.1.3.1 Performing a Typical Installation

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

4.1.3.2 Creating Demonstration Tables Manually

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

```
$ SQLPLUS SYSTEM/MANAGER
SQL> @ORA_SQLPLUS_DEMO:DEMOBLD.SQL
```

4.1.3.3 Deleting Demonstration Tables

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

```
$ SQLPLUS SCOTT/TIGER
SQL> @ORA_SQLPLUS_DEMO:DEMODROP.SQL
```

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

4.2 SQL*Plus Command Line Help

This section describes how to install and remove SQL*Plus command line Help.

See Also: *SQL*Plus User's Guide and Reference*

It contains the following topics:

- Section 4.2.1, "Installing the SQL*Plus Command Line Help"
- Section 4.2.2, "Removing SQL*Plus Command Line Help"

4.2.1 Installing the SQL*Plus Command Line Help

There are two ways to install SQL*Plus command line Help:

- Use Oracle Database Configuration Assistant.

  You can use Oracle Database Configuration Assistant to create `Help` tables when creating a database.

- Run the following command procedure to install the Help facility manually:

  ```
  @ORA_ROOT:[SQLPLUS]HELPINS.COM
  ```

> **Note:** Running this procedure drops any existing command line `Help` tables before creating new tables.

4.2.2 Removing SQL*Plus Command Line Help

You can also run the `ORA_SQLPLUS:HELPDROP.SQL` script in SQL*Plus to manually drop the command line `Help` tables in a schema. For example:

```
$ SQLPLUS SYSTEM/Manager1
SQL> @ORA_ROOT:[SQLPLUS.ADMIN.HELP]HELPDROP.SQL
```

4.3 Using Command-Line SQL*Plus

This section describes how to use SQL*Plus on HP OpenVMS systems. It contains the following topics:

- Section 4.3.1, "Using a System Editor from SQL*Plus"
- Section 4.3.2, "Running Operating System Commands from SQL*Plus"
- Section 4.3.3, "Interrupting SQL*Plus"
- Section 4.3.4, "Using the SPOOL Command"

4.3.1 Using a System Editor from SQL*Plus

If you enter an `ED` or `EDIT` command at the SQL*Plus prompt, then the system starts an operating system editor, such as `EDT` or `TPU`, depending on how the HP OpenVMS `EDIT` symbol is defined.

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

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

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

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

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

SQL*Plus does not delete the temporary file.

4.3.2 Running Operating System Commands from SQL*Plus

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

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

For example, to enter one command:

```
SQL> HOST SHOW DEFAULT
```

or

```
SQL> $ SHOW DEFAULT
```

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

To return to SQL*Plus, enter:

```
$ LOGOUT
```

4.3.3 Interrupting SQL*Plus

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

4.3.4 Using the SPOOL Command

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

```
SQL> SPOOL QUERY.TXT
```

4.4 SQL*Plus Restrictions

This section describes SQL*Plus restrictions. It contains the following topics:

- Section 4.4.1, "Resizing Windows"
- Section 4.4.2, "Return Codes"

4.4.1 Resizing Windows

The default values for SQL*Plus `LINESIZE` and `PAGESIZE` are not automatically adjusted for window size.

4.4.2 Return Codes

When exiting SQL*Plus and returning a status code back to the operating system with `EXIT`, `WHENEVER OSERROR EXIT`, or `WHENEVER SQLERROR EXIT`, HP OpenVMS return codes accept a positive 4 byte number in the range of 0 to 2147483647 (00000000 to 7FFFFFFF in hexadecimal).

5

Configuring Oracle Net Services

This chapter provides conceptual and configuration information about Oracle Net Services in the HP OpenVMS environment.

> **See Also:** *Oracle Database Net Services Administrator's Guide* for detailed information about Oracle Net Services architecture

It contains the following topics:

5.1 Oracle Net Services Configuration Overview

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

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

5.1.1 Centralized Configuration

In a centralized configuration, Oracle Database and Oracle Database Client are located on the same system. This system is not necessarily on a network, and you can access the application through terminals. If you use a centralized configuration, then you can use a simple Oracle Net Services adapter called the *Bequeath adapter*, which requires no Oracle Net Services configuration. However, if you want to use Oracle Shared Servers, then you must configure Oracle Net Services even in centralized configurations.

5.1.2 Client/Server Configuration

In a client/server configuration, Oracle Database resides on a multitasking server system, and the client side of the applications resides on another computer, such as a workstation or personal computer. The client and server are connected by a physical network and communicate through a network protocol such as TCP/IP. In a client/server environment, the Oracle application built with an application development tool makes database requests to the server over the network.

5.1.3 Distributed Database Configuration

In a distributed database configuration, users query separate databases as a single database. The major advantage of a distributed database is that users and applications are not required to know where the data resides. You can query database tables by name, regardless of how the network protocols work together to access the required remote database containing the table. Therefore, Oracle Net Services users can communicate and share database information stored in different locations, on different computers, with different operating systems. Distributed databases enable local administration of data and can reduce network traffic if the data that is accessed most often at a location can be stored locally.

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

5.2 Oracle Net Services Installation

When you install Oracle Net Services on HP OpenVMS, the following protocol adapters are automatically installed:

- TCP/IP
- Bequeath
- IPC
- TCPS

See Also:

- *Oracle Database Installation Guide for HP OpenVMS Itanium*
- The `ORA_RDBMS:READMEVMS.DOC` file

5.3 Oracle Net Services and TNS

Oracle Net Services connects dissimilar networks together and enables client/server transactions to be conducted transparently. A user does not have to know that a

network exists, because Oracle Net Services hides the complexity of system-level interactions by presenting a layer of interconnectivity to the user through its client/server architecture. This layer is called Transparent Network Substrate (TNS).

The transaction proceeds as follows:

1. The client sends a request for data.
2. Oracle Net Services packages the request and sends it to the TNS.
3. TNS routes the packaged request to the server.
4. Oracle Net Services on the server side unpackages the request and sends it to Oracle Database 11*g*.
5. Oracle Database 11*g* processes the request and sends the requested data to Oracle Net Services.
6. Oracle Net Services packages the data and sends it to TNS.
7. TNS routes the data to the client.
8. Oracle Net Services on the client side unpackages the data and sends it to the application.

5.4 Protocol Adapters

Note: This section supplements the information given in *Oracle Database Net Services Administrator's Guide*.

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

- Section 5.4.1, "IPC-Mailbox Protocol"
- Section 5.4.2, "TCP/IP Protocol"
- Section 5.4.3, "BEQ-Bequeath Protocol"
- Section 5.4.4, "Bequeath Listener"

5.4.1 IPC-Mailbox Protocol

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

When configuring the `TNS` listener to listen for mailbox connections, you must specify a `KEY` value in the `LISTENER.ORA` file for the IPC protocol. The listener then creates a mailbox. This mailbox listens for connections and creates a systemwide logical name (the same as the `KEY` value) that translates to this mailbox device. It is through this logical name that clients find the mailbox of the listener.

The Oracle Shared Server must be configured to use only TCP/IP protocol. The IPC protocol cannot be used here.

Syntax

The following fields must be defined:

```
(PROTOCOL=IPC)
```

```
(KEY=IPC logical name)
```

where:

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

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

Example

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

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

5.4.2 TCP/IP Protocol

The TCP/IP protocol adapter provides support for client/server connections using TCP/IP as a protocol. You can turn Oracle Net Services support for TCP/IP on or off by using the options available on the NetConfig configuration screen.

See Also: *Oracle Database Installation Guide for HP OpenVMS Itanium*

Oracle Net Services on HP OpenVMS is developed and certified by using the Hewlett-Packard TCP/IP Services for HP OpenVMS, which is also known as UCX. If you want to use the TCP/IP protocol adapter for Oracle Net Services, then you should have version 5.5 - ECO 1, or later, TCP/IP Services for HP OpenVMS installed. TCP/IP protocol stacks from other vendors may work with Oracle, but customers use these products at their own risk. Any TCP/IP problems that cannot be reproduced by using TCP/IP Services for HP OpenVMS should be referred to the TCP/IP vendor.

Syntax

The following fields must be defined:

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

The following field is optional:

```
(QUEUESIZE=n)
```

In the preceding syntax:

- *PROTOCOL* is the keyword that identifies the specific protocol adapter used.

  For this protocol, the value is TCP. The value can be entered in either uppercase or lowercase.

- *HOST* is the host name or IP address.
- *PORT#* is the TCP/IP port number.
- *Queuesize* is the parameter that increases the queue size. This parameter is optional. If it is not specified, then the default value is 20. If simultaneous connections are made to the listener, then some connection requests may not be received if the listener socket queue size is too small

Example In this example, the TCP/IP connect descriptor specifies a listener on the VMS1 node.

```
(PROTOCOL=TCP)
(HOST=VMS1)
(PORT=1526)
```

5.4.3 BEQ-Bequeath Protocol

Each database that you want to connect with the Bequeath protocol adapter must have a command file named ORASRV_BEQ_sid.COM in ORA_ROOT:[NETWORK.ADMIN]. If you used Oracle Database Configuration Assistant to create the database, or if you upgraded the database, then this file is automatically created. However, if you manually created the database, or if you must re-create this file for any reason, then you must create this command procedure manually by executing ORA_NETWORK:CREATE_ORASRV_BEQ.COM.

5.4.4 Bequeath Listener

On HP OpenVMS, the Bequeath listener is used as a default to provide dedicated server connections for a local client. The Bequeath listener, running as a detached process, creates detached server processes to service clients on the same system by using the Bequeath adapter. This allows Oracle Database to run in a suitably privileged process.

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

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

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

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

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

```
ORA_BEQ_READ_MBX_xxxxxxxxxx_n
```

where:

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

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

This section contains the following topics:

- Section 5.4.4.1, "Starting Up the Bequeath Listener"
- Section 5.4.4.2, "Determining the Status of the Bequeath Listener"
- Section 5.4.4.3, "Shutting Down the Bequeath Listener"
- Section 5.4.4.4, "Problem Resolution"
- Section 5.4.4.5, "Bequeath Listener Privileges"

5.4.4.1 Starting Up the Bequeath Listener

The Bequeath listener starts automatically when the INSORACLE command procedure is run, usually at installation time or during system startup. Unless you decide to run the REMORACLE command, the Bequeath listener should be up and running all the time.

If the Bequeath listener is down and you want to start it, then run the following command:

```
$ BEQLSNR START [n]
```

If you do not provide the optional numeric parameter, then Bequeath listener 0 is started. To start additional Bequeath listeners, provide the listener number when you run the command.

5.4.4.2 Determining the Status of the Bequeath Listener

You can run a status command to determine if the Bequeath listener is up and running. Run the following command:

```
$ BEQLSNR STATUS [n]
```

If you do not provide the optional numeric parameter, then Bequeath listener 0 is queried. To determine the status of any other Bequeath listeners, provide the listener number when you run the command.

5.4.4.3 Shutting Down the Bequeath Listener

To stop the Bequeath listener, run the following command:

```
$ BEQLSNR STOP [n]
```

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

5.4.4.4 Problem Resolution

This section details the steps that you can take to resolve problems with a Bequeath listener.

Collecting Bequeath Client Trace Information:

1. Look for a file in ORA_ROOT:[network.log]BEQ_sid_pid.LOG that corresponds to the time of a client connect/disconnect issue.
2. Make file available to Oracle Support.

Writing Trace Information

The Bequeath listener writes some trace information. Because the output of the detached processes is set to the null device (NL:), usually, you will not see it.

To get the trace information from the Bequeath listener:

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

Changing the Quota for a Server Process That is Created by the Bequeath Listener

To change the process quotas, modify the BEQLSNR.COM file and remove the comment characters for the quota parameter that you want to change. You must stop and then restart the Bequeath listener after modifying this file.

For all ORA-12203 Problems

Ensure that after starting the Bequeath listener, the HP OpenVMS logical name ORA_BEQ has the correct install id as part of its value.

Example (with the Bequeath listener running):

```
$type ora_rootdir:install.id
SERVER_65954DA0

$show log *beq*

(LNM$PROCESS_TABLE)
(SERVER_65954DA0)

"ORA_BEQ" = "65954DA0"

(LNM$JOB_8221BDC0)

(LNM$GROUP_000610)

(LNM$SYSTEM_TABLE)

"ORA_BEQ_READ_MBX_65954DA0_0" = "MBA3737:"
```

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

If you experience this problem, then run the BEQLSNR STATUS command to determine if the Bequeath listener is up and running. If the Bequeath listener does not respond, then run the BEQLSNR STOP command to stop the Bequeath listener and run the BEQLSNR START command to restart it.

> **Note:** Contact Oracle Support if the Bequeath listener does not respond, even after restarting it.

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

Use one of the following solutions:

- Change the ORA_BEQ_TIMEOUT logical name to something greater than 120 seconds. Before running the client program, you must define this logical name in the ORA_NETWORK:BEQLSNR.COM file.
- Define the ORA_BEQ_NUM_OF_LISTENERS logical name to a value between 1 and 10 to increase the capacity, when a number of clients are connecting at the same time to the Bequeath listener.This is the number of Bequeath listener that you must run.

With this method, you can increase the number of connections that the Bequeath listeners can handle concurrently. Each time that a client requests a connection, it randomly picks one of the Bequeath listeners that run to serve it with the connection request. You do not need to stop and restart the Bequeath listener after defining this logical name. This logical name determines the number of Bequeath listeners. However, you must start explicitly each Bequeath listener by running the following command:

```
$ BEQLSNR START n
```

In this command, *n* is an integer that starts from 1 for the first listener and goes up to the value of the ORA_BEQ_NUM_OF_LISTENERS logical name.

5.4.4.5 Bequeath Listener Privileges

The Bequeath listener must have the HP OpenVMS privileges listed in Table 5–1 to be able to perform the associated functions listed in the table.

> **Note:** Before attempting to start the Bequeath listener or the TNS listener, the process that starts the Bequeath listener must have the privileges listed in Table 5–1 or be able to have them set. Refer to Section 5.5.2, "TNS Listener Privileges" for more information about setting TNS listener privileges.

Table 5–1 describes the HP Open VMS privileges and functions of Bequeath TNS listeners.

Table 5–1 Privileges and Functions of Bequeath TNS Listeners

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

5.5 TNS Listener

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

In Oracle Database 11*g*, there is a major change in the way the listener is configured for Oracle Shared Servers. The Oracle Shared Server parameters are not the same as in Oracle8*i*. When you configure for Oracle Shared Server, a request for a dedicated server is no longer handled using the parameters from the LISTENER.ORA file. This now happens as part of the dispatcher registration. The PMON process registers dispatchers with the listener for Oracle Shared Server connections.

Note: The SID_LIST_*listener* section is no longer used to establish dedicated server connections. These are now automatically handled by the listener, which directly uses the ORA_ROOT:[NETWORK.ADMIN]ORASRV_NETV2_*sid*.COM script to launch the dedicated server process. This script is automatically created when a new database or instance is created through Oracle Database Configuration Assistant or scripts provided by Oracle Database Configuration Assistant.

Generic information about the TNS listener and its configuration can be found in the generic Oracle Net Services documentation. This section provides information only about the TNS listener that is specific to HP OpenVMS.

This section contains information about the following topics:

- Section 5.5.1, "LSNRCTL"
- Section 5.5.2, "TNS Listener Privileges"
- Section 5.5.3, "Process Quotas"
- Section 5.5.4, "ORASRV_NETV2_SID Command File"
- Section 5.5.5, "General Connections"

5.5.1 LSNRCTL

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

The main function of the command procedure is to check that the privileges required to start the TNS listener are present. These privileges are covered in the following section. If a LSNRCTL START command is entered and the required privileges are not present, then an error message is displayed and LSNRCTL exits.

Note: Start the TNS listener from the Oracle Account.

Caution: If you enter the LSNRCTL interactive mode by giving the LSNRCTL command without a subcommand, and you have received a warning about inadequate privileges, then do not attempt to start the listener. Depending on the privileges you have, although the listener process may start, it may not function properly.

Do not start the listener from a process that has a User Identification Code (UIC) in the system group, for example, a group less than or equal to MAXSYSGROUP. If you enter a LSNRCTL START command from such a process, then an error message is displayed and LSNRCTL exits. If you enter a LSNRCTL command with no arguments, then you are warned not to start the listener from within the LSNRCTL utility. If the listener is running in a system group, then any server processes it creates will be in the system group. The server is shut down, because it does not let itself run in privileged groups.

5.5.2 TNS Listener Privileges

The process in which the TNS listener runs must have the HP OpenVMS privileges listed in Table 5–1 to be able to perform the associated function.

Note: Before attempting to start the TNS listener, the process that starts the listener must have the privileges in Table 5–1 or be able to have them set. As noted in the preceding section, the LSNRCTL command attempts to set these privileges and warns the user if it was unable to do so.

5.5.3 Process Quotas

Process quotas for the TNS listener and for the server processes created by the TNS listener can be controlled by logical names. The format of the logical name is:

ORA_LSNR_*quotaname*

In this format, *quotaname* can be one of the following:

- ASTLM
- BIOLM
- BYTLM
- CPULM
- DIOLM
- ENQLM
- FILLM
- JTQUOTA
- PGFLQUOTA
- PRCLM
- TQELM
- WSQUOTA
- WSDEFAULT
- WSEXTENT

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

To control the quotas of the processes that the TNS listener creates, specify the logical names in the ORA_NETWORK:TNSLSNR.COM file. This is the command file that runs in the TNS listener process. Statements to define these logical names are in the TNSLSNR.COM file, but these statements are put in comments.

If, for example, a very large SGA requires that server processes have larger quotas, then you can activate the appropriate logical name definition in TNSLSNR.COM file by removing the "!" after the "$" and specifying the quota value.

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

5.5.4 ORASRV_NETV2_*SID* Command File

The ORASRV_NETV2_*SID*.COM file is automatically created for each *SID* during creation of a database instance.

If an Oracle Shared Server is not being used, then the behavior is the same as seen in earlier releases. The PROGRAM parameter should point to this script in the LISTENER.ORA file. For example:

```
(SID_LIST_LISTENER =
 (SID_DESC =
    (SID_NAME = PROD)
    (PROGRAM= MY_DISK:[home.NETWORK.ADMIN]ORASRV_NETV2_PROD.COM)
  )
)
```

When the TNS listener starts a dedicated server process, it extracts the value of the PROGRAM parameter from the LISTENER.ORA file.

In an Oracle Shared Server configuration, the TNS listener need not contain the SID_LIST_*listener* section mentioned earlier. The Oracle Shared Server dispatchers register with the TNS listener directly. These dispatchers also specify the command procedure to run for a dedicated procedure.

This command procedure is currently set to the ORA_ ROOT:[NETWORK.ADMIN]ORASRV_NETV2_*SID*.COM, which is created automatically. The location and format of the name of this file cannot be changed. A SID_LIST section in the LISTENER.ORA file that points to the same or different script is completely ignored.

5.5.5 General Connections

Ensure that the Oracle Net Services task file defines any logical names used by the INIT.ORA parameters USER_DUMP_DEST and BACKGROUND_DUMP_DEST (if defined).

5.5.6 Trace Information for TCPIP connection

If the TCPIP connection error is unsuccessful, then check for the following file:

```
ORA_ROOT:[NETWORK.LOG]NETV2_sid_pid.LOG
```

This file matches the time of the error. Provide this file to Oracle Support.

5.6 Advanced Security Option

This section provides information that is specific to the current release of Advanced Security Option (ASO) for Security and Single Sign-On.

Note: A separate license is required to use ASO.

This section covers the following topics:

- Section 5.6.1, "Manual Steps for Authentication Adapters"

- Section 5.6.2, "Usage Notes for Authentication Adapters"

5.6.1 Manual Steps for Authentication Adapters

Set the following parameters in the local INIT.ORA file of the database server:

```
remote_os_authent = false
os_authent_prefix = ""
```

For Kerberos5 Adapter

The KRB.CONF file is required on the client side. This configuration file specifies the default realm of the client and maps all known realms to Key Distribution Centers (KDCs).

The following files are required on the server side:

- KRB.REALMS: This file maps host names and domains into realms.
- V5SRVTAB: This file contains a key that the KDC uses to encrypt a service ticket for the client.

The location of these files must be specified by using corresponding parameters in the SQLNET.ORA file.

In addition, the Oracle Net Services client also creates a credential cache file, whose location must be specified in the SQLNET.ORA file on the client side.

The following is an example of the parameters in the SQLNET.ORA file for an installation that can act as both client and server:

```
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=ORACLE
SQLNET.AUTHENTICATION_SERVICES = (BEQ,KERBEROS5)
SQLNET.KERBEROS5_KEYTAB = DISK:[TST901.NETWORK.ETC]V5SRVTAB.
SQLNET.KERBEROS5_CONF = DISK:[TST901.NETWORK.KRB5]KRB.CONF
SQLNET.KERBEROS5_REALMS = DISK:[TST901.NETWORK.KRB5]KRB.REALMS
SQLNET.KERBEROS5_CC_NAME = DISK:[TST901.NETWORK.CCACHE]CCFILE.DAT
```

5.6.2 Usage Notes for Authentication Adapters

This section covers the usage notes for authentication adapters.

General Information

Include the following line in the SQLNET.ORA file:

```
SQLNET.AUTHENTICATION_SERVICES=(NONE)
```

The listener should not participate in the authentication service.

It is recommended that you always include BEQ as one of the authentication services in the SQLNET.ORA file. For example:

```
SQLNET.AUTHENTICATION_SERVICES=(BEQ,KERBEROS5)
```

In this way, connections within the server system through the default Bequeath adapter do not have to go through authentication. This is especially important during database startups and shutdowns.

Kerberos 5

1. Ensure that the clock skew between the client system and the system running the KDC is less than one minute.

2. Oracle Database client and server processes use the Coordinated Universal Time (UTC) format (time elapsed since 00:00:00 Jan. 1, 1970 in records). Ensure that the system is set to the correct time zone in terms of deviation from Greenwich Mean Time (GMT). Otherwise, the following error message is written to the Oracle Net Services trace file:

   ```
   Clock skew too great
   ```

3. Ensure that the value of the SQLNET.AUTHENTICATION_KERBEROS5_SERVICE parameter that you specify in the SQLNET.ORA file matches exactly with the value specified in the KDC. This value is case-sensitive.

5.7 Oracle Net Services Configuration Files

This section describes the files that you can use to configure Oracle Net Services products.

See Also: *Oracle Database Net Services Administrator's Guide*

The default directory for Oracle Net Services configuration files is ORA_ROOT:[NETWORK.ADMIN] or TNS_ADMIN.

Oracle Net Services searches for configuration files in the following order:

1. For the SQLNET.ORA file, the current working directory from where an application is run
2. The directory specified by the TNS_ADMIN logical name, if it is set

For each system-level configuration file, users may have a corresponding local private configuration file (stored in the user's home directory). The settings in the local file override the settings in the system-level file.

Table 5–2 lists the system-level configuration files and the corresponding local configuration files.

Table 5–2 Oracle Net Configuration Files

System-Level Configuration Files	Local Configuration Files
SQLNET.ORA	TNS_ADMIN:SQLNET.ORA
TNSNAMES.ORA	TNS_ADMIN:TNSNAMES.ORA

5.8 Configuring Oracle Net Services Protocol Support

Oracle Net Services release 11*g* on HP OpenVMS supports the following protocols:

- Bequeath (BEQ)
- IPC
- TCP/IP

Before installing the TCP/IP protocol support, you must install and configure the required operating system software. BEQ and IPC protocol support does not have any specific operating system requirements.

On HP OpenVMS, the Oracle Shared Server must be configured to use only TCP/IP protocol.

See Also: *Oracle Database Installation Guide for HP OpenVMS Itanium* for more information about Oracle Net Services protocol support

5.8.1 ADDRESS Specification

IPC and TCP/IP protocol support have a protocol-specific ADDRESS specification that is used for Oracle Net Services configuration files and for the `DISPATCHERS` initialization parameter in the `init`*`sid`*`.ora` file.

Table 5–3 shows the ADDRESS specifications for each supported protocol.

Table 5–3 ADDRESS Specification Summary

Supported Protocol	ADDRESS Specification

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

TCP/IP	`(ADDRESS =`
	`(PROTOCOL=TCP)`
	`(HOST=hostname)`
	`(PORT=port)`
	`)`

5.9 BEQ Protocol Support

The Bequeath (BEQ) protocol support is both a communications mechanism and a process-spawning mechanism. To use the BEQ protocol support, the client and the server must be on the same system. A network service name can be specified directly by the user at the command line or on the Login screen. It can also be specified indirectly by using a logical name, such as `ORA_DFLT_HOSTSTR`.

If a network service name is not specified, then the BEQ protocol support is used. In this case, the BEQ protocol support always uses a dedicated server and the shared server model is never used. This dedicated server is started automatically by the BEQ protocol, which waits for the server process to start and attach to an existing System Global Area (SGA). If the startup of the server process is successful, then the BEQ protocol support provides interprocess communication through HP OpenVMS mailboxes.

An important feature of the BEQ protocol support is that it does not require a listener for its operation. The protocol support is linked to the client tools and directly starts its own server process without outside interaction. However, you can use the BEQ protocol support only when the client program and Oracle Database 11*g* are installed on the same system. The BEQ protocol support is always installed and always linked to all client tools and to the Oracle Database 11*g* server.

5.10 IPC Protocol Support

The IPC protocol support is similar to the BEQ protocol support in that it can be used only when the client program and the Oracle Database 11*g* server are installed on the same system. The IPC protocol support requires a listener for its operation. The IPC

protocol support is always installed and always linked to all client tools and to Oracle Database 11*g*

Specifying an IPC ADDRESS

The IPC protocol support connection parameters are part of the ADDRESS keyword-value pair. The ADDRESS is commonly part of a larger construct, such as a connect descriptor or configuration file. You can enter the following parameters in any order:

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

Table 5–4 describes the syntax for IPC protocol connection parameters.

Table 5–4 Syntax for IPC Protocol Connection Parameters

Parameter	Description
PROTOCOL	IPC protocol support to be used The value is IPC. It is not case-sensitive.
KEY	Service name of database or system identifier (*SID*)

Example 5–1 shows a sample IPC ADDRESS.

Example 5–1 IPC ADDRESS Specifying a Client

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

5.11 TCP/IP Protocol Support

On HP OpenVMS, the default `tnslnr` port is 1521.

5.11.1 Specifying a TCP/IP ADDRESS

The TCP/IP protocol connection parameters are part of the ADDRESS keyword-value pair. The ADDRESS is commonly part of a larger construct such as a connect descriptor or configuration file. You can enter the parameters in any order:

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

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

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

Parameter	Description
PROTOCOL	The protocol support to be used The value is TCP. It is not case-sensitive.
HOST	The host name or the host IP address

Table 5–5 (Cont.) Syntax for TCP/IP Protocol Connection Parameters

Parameter	Description
PORT	The TCP/IP port

Example 5–2 shows a sample TCP/IP ADDRESS.

Example 5–2 TCP/IP ADDRESS Specifying a Client

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

You can specify the last field by name as follows:

```
PORT=listener_name
```

6

Using Oracle Precompilers and the Oracle Call Interface

This chapter describes how to use Oracle Precompilers and the Oracle Call Interface.

It contains the following topics:

6.1 Overview of Oracle Precompilers

Oracle Precompilers are application development tools used to combine SQL statements from an Oracle Database with programs written in a high-level language. Oracle Precompilers are compatible with ANSI SQL and develop open, customized applications that run with the Oracle Database or any other ANSI SQL database management system.

> **See Also:** *Programmer's Guide to the Oracle Precompilers* for general information about Oracle Precompilers and interface features

6.1.1 Precompiler Configuration Files

System configuration files for the Oracle Precompilers are located in the `ORA_ROOT:[PRECOMP.ADMIN]` directory.

Table 6–1 lists the names of the system configuration files for each precompiler. These files are currently empty. Their purpose is to specify command-line parameters, such as the `include` directories.

Table 6–1 System Configuration Files for Oracle Precompilers

Product	Configuration File
Pro*C/C++	`PCSCFG.CFG`
Pro*COBOL	`PCBCFG.CFG`
Pro*FORTRAN	`PCCFOR.CFG`
Object Type Translator	`OTTCFG.CFG`

6.1.2 Precompiler Executables

Table 6–2 lists products and their corresponding executable names as well as the HP OpenVMS symbols associated with them. These images are found in the `ORA_ROOT:[BIN]` directory.

6.1.2.1 Precompiler README files

Precompiler `README` files for the various languages are located in `ORA_ROOT:[PRECOMP.DOC.`*`LANGUAGE`*`]`.

Table 6–2 lists precompiler products, executable names and HP OpenVMS symbols.

Table 6–2 Executable Names and HP OpenVMS Symbols for Precompiler Products

Product	Executable	HP OpenVMS Symbol
Pro*C/C++	`PROC.EXE`	PROC
Pro*COBOL	`PROCOB.EXE`	PROCOB
Pro*FORTRAN	`PROFOR.EXE`	PROFOR
Object Type Translator	`OTT.COM`	OTT

6.1.3 Precompiler README Files

Table 6–3 lists the location of the precompiler `README` files. The `README` files describe changes made to the precompiler since the last release.

Table 6–3 Location of Precompiler README Files

Precompiler	README File
Pro*C/C++	`ORA_ROOT:[PRECOMP.DOC.PROC]README.DOC`
Pro*COBOL	`ORA_ROOT:[PRECOMP.DOC.PROCOB2]README.DOC`
Pro*FORTRAN	`ORA_ROOT:[PRECOMP.DOC.PRO1X]README.TXT`

6.1.4 Issues Common to All Precompilers

The following issues are common to all precompilers.

Conversion for Uppercase to Lowercase

In languages other than the C programming language, the compiler converts an uppercase function or subprogram name to lowercase. This can cause the following error message to be displayed:

```
No such user exists
```

If this error message is displayed, then verify that the function or subprogram name in the option file matches the case used in the `IAPXTB` table.

Vendor Debugger Programs

Precompilers and vendor-supplied debuggers can be incompatible. Oracle does not guarantee that a program run using a debugger performs the same way when it is run without the debugger.

Values of the IRECLEN and ORECLEN Parameters

The `IRECLEN` and `ORECLEN` parameters do not have maximum values.

6.1.5 Static and Dynamic Linking

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

With dynamic linking, the executing code is partly stored in the executable program and partly stored in libraries that are loaded with the application at run time, or later, after the program starts, when referenced. These libraries are called sharable libraries or dynamic libraries. The benefits of dynamic linking are:

- Reduced disk space requirements: More than one application or invocation of the same application can use the same dynamic library.
- Reduced main memory requirements: The same dynamic library image is loaded into main memory only once and it can be shared by more than one application, if the library is installed as a shared image.

6.1.6 Client Shared and Static Libraries

The client shared and static libraries are located in the `ORA_ROOT:[LIB32]` directory. If you use the Oracle provided installer or link scripts to link an application, then the client shared library is linked by default. The necessary logical name for the client shared library is defined by default when the client environment is set up.

The client shared library is created automatically during installation. If you must re-create it, then perform the following:

1. Exit all client applications that use the client shared library, including all Oracle Database client applications.
2. Log in as the `oracle` user, and enter the following command:

   ```
   @ORA_ROOT:[BIN]RELINK CLIENT_SHAREDLIB
   ```

If an application links with the client shared library and needs to be installed as a known image, HP OpenVMS requires that an executive mode logical name point to the shared library.

If there is only one Oracle Installation on the system, add the following command to the system startup file:

```
$ DEFINE/SYSTEM/EXEC LIBCLNTSH full_Path/LIBCLNTSH.SO
```

Do not use concealed logical names, including ORA_ROOT, in the *full_path* specification.

If you have multiple Oracle installations on the system, you must uniquely identify the client shared library for the installation in question. Do the following:

1. Copy LIBCLNTSH.SO to LIBCLNTSH_*unique_id*.SO, where the *unique_id* is a string not common to any other Oracle installation on the system.
2. Install the shared library LIBCLNTSH_*unique_id*.SO. Refer to ORA_ROOT:[000000]INSORACLE.COM for the correct syntax.
3. Define the logical name LIBCLNTSH_*unique_id* as follows:

   ```
   $ DEFINE/SYSTEM/EXEC LIBCLNTSH_unique_id full_path/LIBCLNTSH_unique_id.SO
   ```

   Again, do not use concealed logical names in the *full_path*.
4. Link your application with this uniquely named client shared library. you must modify the file ORA_ROOT:[RDBMS]ORA_CLIENT.OPT to do this

Steps 2 and 3 the previous procedure should be added to the system startup command file.

6.2 Precompiling

You start the precompilers and Object Type Translators by using the HP OpenVMS symbols specified in Table 6–2.

This section covers the following aspects of precompiling:

- Section 6.2.1, "Syntax"
- Section 6.2.2, "Guidelines and Restrictions"

6.2.1 Syntax

The syntax of the command for precompiling source files is as follows:

```
$ VMS_symbol INAME=filename OPTION=value ...
```

In this syntax:

- *VMS_symbol* is the HP OpenVMS symbol for the precompiler
- *filename* is the name of the source file you want to precompile
- *OPTION* is the precompiling option available for the Oracle Precompilers program You can supply any number of option-value pairs, separated by a space
- *value* is the value of the option specified

Example 6–1 Syntax for Precompiling

```
$ 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 this option is not provided, then the path to the directory in which the include files are distributed is taken as the default.

You can display a list of options and their values (if you have an Oracle instance running) by entering the required symbol name. For example:

```
$ PROFOR
```

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

6.2.2 Guidelines and Restrictions

The following guidelines and restrictions apply to precompiling:

Using the HP OpenVMS Debugger

Precompiler programs can be run with the HP 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 HP OpenVMS event flags in the source code, then ensure that none of them are numbered 1 through 18 before compiling the code for use with Oracle Database. Event flags 1 through 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, ORACA, or both. Oracle recommends that you insert the following definition in one module to produce a defining declaration of the SQLCA structure:

```
#define SQLCA_STORAGE_CLASS GLOBALDEF
```

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

```
#define SQLCA_STORAGE_CLASS GLOBALREF
```

This line must precede inclusion of SQLCA.H.

6.3 Compiling

You must ensure that the conditions described in the following are met when using the precompilers listed in this section:

- Section 6.3.1, "Compiler Options Used to Compile Oracle Database 11g"
- Section 6.3.2, "Floating Point Format"
- Section 6.3.3, "Pro*COBOL"

6.3.1 Compiler Options Used to Compile Oracle Database 11*g*

Oracle Database 11*g* is compiled with as few deviations from the default C compiler options as possible and with minimal use of pragma statements.

For the HP C compiler on HP OpenVMS, the compilation options are as follows:

```
/DECC /NOSTANDARD /DEBUG=TRACE /PREFIX_LIBRARY_ENTRIES=ALL_ENTRIES
/GRANULARITY=LONGWORD /NAMES=AS_IS /FLOAT=IEEE
/IEEE_MODE=DENORM_RESULTS /EXTERN_MODEL=STRICT_REFDEF /NOANSI_ALIAS
```

If you compile the code with `/DEBUG=TRACE`, then line numbers in the modules are displayed, as required, in Oracle Database 11*g* stack trace listings.

6.3.2 Floating Point Format

Oracle Database 11*g* is compiled with the IEEE floating point format supported by the C compiler. The conversion routines within Oracle Database 11*g* translate operating system-specific floating point numbers into Oracle Database 11*g* internal floating point representation.

This is a change from Oracle Database 9*i* release 2 for HP OpenVMS. If you had an application that depended on non-IEEE defaults, then you may need to recompile.

6.3.2.1 Application Compatibility for Floating Point Format

With Oracle Database 11*g* for HP OpenVMS, the floating point format supported by Oracle code is the IEEE floating point format.

Earlier releases of the product supported the default native floating format of the C compiler, which was the `F` float for single and `G` float for double.

6.3.3 Pro*COBOL

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

6.4 Linking

Use the following command procedures to link object files:

- `LNPRO`*`language`*`.COM`

  `LNPRO`*`language`*`.COM` is the standard, recommended linking method.

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

- `LNOCI.COM` to link OCI programs that are not written in the C programming language
- `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`, then use a command syntax similar to that found in the required `LNPRO`*`language`*`.COM` script.

- `LNPROC == "@ora_proc:lnproc"`
- `LNPROCOB == "@ora_procob:lnprocob"`

- `LNPROCXX == "@ora_proc:lnprocxx"`
- `LNPROFOR == "@ora_profor:lnprofor"`

This section discusses the following aspects of linking:

- Section 6.4.1, "Syntax"
- Section 6.4.2, "Linking Precautions"
- Section 6.4.3, "Guidelines for Linking"

6.4.1 Syntax

To link compiled `PRO`*`language`* object files, use the `LNPRO`*`language`* symbol.

Table 6–4 provides a description of each argument.

Table 6–4 Linking Precompiled Programs

Argument	Description
language	Abbreviation for the programming language you are using For example: C, COB, or FOR
executable	Name of the executable image to be created The file name extension is optional.
objectfilelist	Comma-delimited list of object files and libraries If this list is longer than one line, then use the continuation character, the dash sign (-). There are no spaces in this specification.
options	List of options with no separators needed: `D` links with the HP OpenVMS `DEBUG` utility. `F` produces a full map. `M` creates a link map. `X` produces a link map with cross-references.

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

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

6.4.2 Linking Precautions

Oracle Database 11*g* is compiled with the IEEE floating point format supported by the C compiler.

Applications that were compiled with earlier releases of the Oracle Database 11*g* for HP OpenVMS should not be directly linked with the static or dynamic libraries that are provided with the current release. Although they may link correctly, run-time results may be unpredictable. Oracle recommends that all such applications be either recompiled or continue to be run from a 10.2 client environment connecting to Oracle Database 11*g* through Oracle Net Services.

6.4.3 Guidelines for Linking

Apply the guidelines defined in this section when using link scripts.

Using the Demonstrations

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

Before running the PROC demonstrations, ensure that you define the following environment setting:

```
$ DEFINE SYS DECC$LIBRARY_INCLUDE
```

Compatibility with ANSI Standard Compilers

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

Linking Sharable Images with LOUTL.COM

You may link a sharable image against Oracle Database 11*g* code using the D option with LOUTL.COM or one of the LNPRO*.COM link scripts that internally calls LOUTL.COM.

You may want to install the sharable image in system memory with a command similar to the following:

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

To avoid receiving an error when you link the main program, include the sharable image in the link list.

Watching the Link Command Passed to LOUTL

LOUTL looks for the symbol SHOW_LINK_COMMAND, which lets you 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, then LOUTL displays the link command. If this symbol is undefined, then LOUTL issues the link command silently.

Using LNK$LIBRARY When Linking Against Oracle

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

6.5 Pro*C/C++ Precompiler

Before you use the Pro*C/C++ precompiler, verify that the correct version of the operating system compiler is properly installed.

See Also:

- *Oracle Database Installation Guide for HP OpenVMS Itanium* for information about the required compiler versions
- *Pro*C/C++ Programmer's Guide* for information about the Pro*C/C++ precompiler and interface features

This section discusses the following topics:

- Section 6.5.1, "Pro*C/C++ Demonstrations"
- Section 6.5.2, "Pro*C/C++ User Programs"

- Section 6.5.3, "Operating System (HP OpenVMS) Header Files"

6.5.1 Pro*C/C++ Demonstrations

Demonstrations are provided to show the features of the Pro*C/C++ precompiler. There are three types of demonstrations: C, C++, and Object programs. All the demonstrations are located in the ORA_ROOT:[PRECOMP.DEMO.PROC] directory. By default, all programs are dynamically linked with the client shared library.

To run, the programs require the demonstration tables created by the ORA_ROOT:[SQLPLUS.DEMO]DEMOBLD.SQL script to exist in the SCOTT schema with the password TIGER.

> **Note:** You must unlock the SCOTT account and set the password before creating the demonstrations.

For example, the following are the steps to precompile, compile, and link the sample1 demonstration program:

The DBA of the site must install Oracle Database Sample Schemas and unlock SCOTT/TIGER. For example:

```
$ @disk:[oracle_home_directory]ORAUSER sid
$ sqlplus / as sysdba
```

> **Note:** The following is for illustrative purposes only. Contact the DBA of the site for information about the required security settings.

```
SQL> ALTER USER SCOTT ACCOUNT UNLOCK;
SQL> ALTER USER SCOTT IDENTIFIED BY TIGER;
SQL> EXIT
$ SET DEFAULT ORA_ROOT:[PRECOMP.DEMO.PROC]
```

> **Note:** The following is for illustrative purposes only. Contact the system manager of the site for information about the required C programming language compiler header location.

```
$ PROC INAME=SAMPLE1 INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")')
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG -
/NAMES=AS_IS /FLOAT=IEEE /INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC] SAMPLE1.C
$ LNPROC SAMPLE1
$ RUN SAMPLE1
7934
0
```

Use similar commands for the demonstration programs SAMPLE2, 3, 6, 7, 8, 12, SQLVCP.PC and SCDEMO2.PC. This procedure does not require data entry.

To create the Proc*C SAMPLE4 demonstration, additional parameters and data entry are required as follows:

Note: The following is for illustrative purposes only. Contact the system manager of the site for information about the required C programming language compiler definitions.

```
$ DEFINE SYS DECC$LIBRARY_INCLUDE
$ PROC INAME=SAMPLE4 INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")',ORA_
ROOT:[RDBMS.PUBLIC])
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG -
/NAMES=AS_IS /FLOAT=IEEE -
/INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC]) SAMPLE4.C
$ LNPROC SAMPLE4
$ RUN SAMPLE4
y
l
i
dd
sample4.pc
l
r
dd
tsamp4.pc
d
dd
l
q
```

Note: SAMPLE5.PC is not supported.

Some demonstrations require you to run a SQL script that is located in the ORA_ROOT:[PRECOMP.DEMO.SQL] directory. If you do not run the script, then a message requesting you to run it is displayed. For example, to create the SAMPLE9 demonstration program and run the required ORA_ROOT:[PRECOMP.DEMO.SQL]CALLDEMO.SQL script, run the following commands:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]CALLDEMO.SQL
$ PROC INAME=SAMPLE9 INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")') SQLCHECK=FULL
USER=SCOTT/TIGER
$ CC /DECC/NOSTANDARD/DEBUG=TRACE/OPTIMIZE/PREFIX=ALL/GRAN=LONG/NAMES=AS_IS -
/FLOAT=IEEE/INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC] SAMPLE9.C
$ LNPROC SAMPLE9
$ RUN SAMPLE9
30
```

Running the SAMPLE10 demonstration requires additional data entry as follows:

```
$ RUN SAMPLE10
scott
tiger
select * from dept;
select * from emp;
exit
```

Running the SAMPLE11 demonstration requires additional SQL statements, parameters, and data entry as follows:

```
$ SQLPLUS/NOLOG
```

```
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]SAMPLE11.SQL
$ PROC INAME=SAMPLE11 INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")') SQLCHECK=FULL
USER=SCOTT/TIGER
$ CC /DECC/NOSTANDARD/DEBUG=TRACE/OPTIMIZE/PREFIX=ALL/GRAN=LONG -
/NAMES=AS_IS/FLOAT=IEEE/INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC] SAMPLE11.C
$ LNPROC SAMPLE11
$ RUN SAMPLE11
10
20
30
40
0
```

Running the `ANSIDYN1` demonstration requires additional parameters and data entry:

```
$ PROC INAME=ANSIDYN1 INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")') MODE=ANSI
$ CC /DECC/NOSTANDARD/DEBUG=TRACE/OPTIMIZE/PREFIX=ALL/GRAN=LONG -
/NAMES=AS_IS/FLOAT=IEEE/INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC] ANSIDYN1.C
$ LNPROC ANSIDYN1
$ RUN ANSIDYN1
scott
tiger
SELECT empno, ename, mgr FROM emp;
SELECT deptno, dname FROM dept;
EXIT;
```

Running the `ANSIDYN2` demonstration requires additional parameters and data entry as follows:

```
$ PROC INAME=ANSIDYN2 INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")') DYNAMIC=ANSI
$ CC /DECC/NOSTANDARD/DEBUG=TRACE/OPTIMIZE/PREFIX=ALL/GRAN=LONG -
/NAMES=AS_IS/FLOAT=IEEE/INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC] ANSIDYN2.C
$ LNPROC ANSIDYN2
$ RUN ANSIDYN2
scott
tiger
SELECT empno, ename, mgr FROM emp;
1
SELECT deptno, dname FROM dept;
1
EXIT;
10
20
0
```

Running the `CV_DEMO` demonstration requires additional SQL statements, parameters, and data entry as follows:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]CV_DEMO.SQL
$ PROC INAME=CV_DEMO USER=SCOTT/TIGER SQLCHECK=FULL -
INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")',ORA_ROOT:[RDBMS.PUBLIC])
$ CC /DECC/NOSTANDARD/DEBUG=TRACE/OPTIMIZE/PREFIX=ALL/GRAN=LONG -
/NAMES=AS_IS/FLOAT=IEEE/INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC] CV_DEMO.C
$ LNPROC CV_DEMO
$ RUN CV_DEMO
```

Running the `LOBDEMO1` demonstration requires additional SQL statements, parameters, and data entry as follows:

```
$ SQLPLUS/NOLOG
```

```
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]LOBDEMO1.SQL
$ PROC INAME=LOBDEMO1 SQLCHECK=FULL USER=SCOTT/TIGER -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")',ORA_ROOT:[RDBMS.PUBLIC])
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS
-
/FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC],ora_root:[rdbms.public])
LOBDEMO1.C
$ LNPROC LOBDEMO1
$ RUN LOBDEMO1
L
G
555001212
I
123456789
John Doe
A
123456789
2
Q
```

Alternatively, `LOBDEMO1.PC` can be precompiled with by setting `cpool` to the values `yes` as follows:

```
$ PROC INAME=LOBDEMO1 SQLCHECK=FULL USER=SCOTT/TIGER CPOOL=YES -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")',ORA_ROOT:[RDBMS.PUBLIC])
```

The remainder of the demonstration is the same as shown earlier.

Running the `ORACA` demonstration requires additional SQL statements and data entry as follows:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]ORACATST.SQL
$ PROC INAME=ORACA INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")')
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG -
/NAMES=AS_IS /FLOAT=IEEE /INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC] ORACA.C
$ LNPROC ORACA
$ RUN ORACA
10
```

Running the `SCDEMO1` demonstration requires additional parameters and data entry as follows:

```
$ PROC INAME=SCDEMO1 INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")')
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG -
/NAMES=AS_IS /FLOAT=IEEE /INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC] SCDEMO1.C
$ LNPROC SCDEMO1
$ SCDEMO1 :== $ORA_ROOT:[PRECOMP.DEMO.PROC]SCDEMO1.EXE
$ SCDEMO1 SCOTT/TIGER
SELECT ENAME, JOB FROM EMP
1
1
y
4
n
EXIT
```

The `CPDEMO1.PC` and `CPDEMO2.PC` demonstrations are not supported.

To precompile, compile, and link the C++ `CPPDEMO1.PC` demonstration program, run the following commands:

```
$ SET DEFAULT ORA_ROOT:[PRECOMP.DEMO.PROC]
```

> **Note:** The following is for illustrative purposes only. Contact the system manager of the site for information about the required header locations for the C and C++ programming language compilers.

```
$ PROC INAME=CPPDEMO1  DEFINE=_RWSTD_USE_CONFIG CODE=CPP -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_
INCLUDE")',SYS$COMMON:[CXX$LIB.REFERENCE.CXXL$ANSI_DEF])
$ CXX /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS /FLOAT=IEEE -
/INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) CPPDEMO1.C/DEFINE=(_RWSTD_USE_CONFIG)
$ LNPROC CPPDEMO1 CPPDEMO1 CPP
$ RUN CPPDEMO1
7369
7499
0
```

To create the Pro*C++ `CPPDEMO2.PC` demonstration program which also includes `EMPCLASS.PC`, run commands similar to those of `CPPDEMO1.PC`, but without data entry, as follows:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]CPPDEMO2.SQL
$ PROC INAME=EMPCLASS CODE=CPP SQLCHECK=FULL USER=SCOTT/TIGER -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_
INCLUDE")',SYS$COMMON:[CXX$LIB.REFERENCE.CXXL$ANSI_DEF])
$ CXX /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG -
/NAMES=AS_IS /FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) EMPCLASS.C
$ PROC INAME=CPPDEMO2 CODE=CPP -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_
INCLUDE")',SYS$COMMON:[CXX$LIB.REFERENCE.CXXL$ANSI_DEF])
$ CXX /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS -
/FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) CPPDEMO2.C
$ LNPROC CPPDEMO2 CPPDEMO2,EMPCLASS CPP
$ RUN CPPDEMO2
```

To create the Pro*C++ `CPPDEMO3.PC` demonstration program, run commands similar to those of `CPPDEMO1.PC`. This procedure does not require data entry.

To precompile, compile, and link the OTT `COLDEMO1.PC` demonstration program, run the following commands:

```
$ SET DEFAULT ORA_ROOT:[PRECOMP.DEMO.SQL]
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]COLDEMO1.SQL
$ SET DEFAULT ORA_ROOT:[PRECOMP.DEMO.PROC]
$ @ORA_ROOT:[JDBC]JDBC_SETUP_JDK12.COM
$ OTT COLDEMO1 CODE=C USERID=SCOTT/TIGER INTYPE=COLDEMO1.TYP OUTTYPE=COLDEMO1O.TYP
HFILE=COLDEMO1.H
$ PROC INAME=COLDEMO1 INTY=COLDEMO1O.TYP -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")',ORA_ROOT:[RDBMS.PUBLIC])
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS
-
/FLOAT=IEEE /INCLUDE=([],ORA_ROOT:[PRECOMP.PUBLIC],ORA_ROOT:[RDBMS.PUBLIC])
COLDEMO1.C
$ LNPROC COLDEMO1
$ RUN COLDEMO1
A
```

```
Q
```

Optionally, COLDEMO1.PC may also be precompiled with "cpool=yes" as follows:

```
$ PROC INAME=COLDEMO1 INTY=COLDEMO1O.TYP CPOOL=YES -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")',ORA_ROOT:[RDBMS.PUBLIC])
```

To run the Pro*C NAVDEMO1.PC, OBJDEMO1.PC OTT demonstrations, run commands similar to COLDEMO1.PC. This procedure does not require data entry.

To precompile, compile, and link the OTT COLDEMO2.PC demonstration program, run the following commands:

```
$ SET DEFAULT ORA_ROOT:[PRECOMP.DEMO.SQL]
$ SQLPLUS/NOLOG
SQL> CONNECT SCOTT/TIGER
SQL> DROP TABLE COUNTY_TBL;
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]COLDEMO2.SQL
$ SET DEFAULT ORA_ROOT:[PRECOMP.DEMO.PROC]
$ @ORA_ROOT:[JDBC]JDBC_SETUP_JDK12.COM
$ OTT COLDEMO2 CODE=C USERID=SCOTT/TIGER INTYPE=COLDEMO2.TYP OUTTYPE=COLDEMO2O.TYP
HFILE=COLDEMO2.H
$ PROC INAME=COLDEMO2 INTY=COLDEMO2O.TYP -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")',ORA_ROOT:[RDBMS.PUBLIC])
$ CC /DECC /NOSTANDARD /DEBUG=TRACE -
/OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS /FLOAT=IEEE -
/INCLUDE=([],ORA_ROOT:[PRECOMP.PUBLIC],ORA_ROOT:[RDBMS.PUBLIC]) COLDEMO2.C
$ LNPROC COLDEMO2
$ RUN COLDEMO2
Y
1999
A
Q
```

6.5.2 Pro*C/C++ User Programs

To create and run a program, enter commands similar to the following:

```
$ PROC INAME=objfile1 INCLUDE=('f$trnlnm("DECC$LIBRARY_INCLUDE")')
$ PROC INAME=objfile2 INCLUDE=('f$trnlnm("DECC$LIBRARY_INCLUDE")')
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS
-
/FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) objfile1.c
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS
-
/FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) objfile2.c
$ LNPROC OBJFILE1 OBJFILE1,OBJFILE2
$ RUN OBJFILE1
```

In this example:

- OBJFILEN is the C source file for the program
- the first LNPROC parameter is the executable program

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

- For C source, linked with the client shared library, enter the following commands:

```
$ PROC INAME=MYPROG INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")')
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG -
```

```
/NAMES=AS_IS /FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) MYPROG.C
$ LNPROC MYPROG
```

- For C source, linked with the client object library, enter the following commands:

```
$ PROC INAME=MYPROG INCLUDE=('F$TRNLNM("DECC$LIBRARY_INCLUDE")')
$ CC /DECC /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG -
/NAMES=AS_IS /FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) MYPROG.C
$ DEFINE ORA_OLB ORA_ROOT:[LIB32],ORA_RDBMS
$ DEFINE ORA_UTIL ORA_OLB
$ LOUTL MYPROG MYPROG MYPROG NS
```

- For C++ source, linked with the client shared library, enter the following commands:

```
$ PROC INAME=myprog CODE=CPP -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_
INCLUDE")',SYS$COMMON:[CXX$LIB.REFERENCE.CXXL$ANSI_DEF])
$ CXX /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS -
/FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) MYPROG.C
$ LNPROC MYPROG MYPROG CPP
```

- For C++ source, linked with the client object library, enter the following commands:

```
$ PROC INAME=MYPROG CODE=CPP -
/INCLUDE=('F$TRNLNM("DECC$LIBRARY_
INCLUDE")',SYS$COMMON:[CXX$LIB.REFERENCE.CXXL$ANSI_DEF])
$ CXX /NOSTANDARD /DEBUG=TRACE /OPTIMIZE /PREFIX=ALL /GRAN=LONG /NAMES=AS_IS -
/FLOAT=IEEE /INCLUDE=(ORA_ROOT:[PRECOMP.PUBLIC]) MYPROG.C
$ DEFINE ORA_OLB ORA_ROOT:[LIB32],ORA_RDBMS
$ DEFINE ORA_UTIL ORA_OLB
$ LOUTL MYPROG MYPROG MYPROG NS CPP
```

6.5.3 Operating System (HP OpenVMS) Header Files

If you want to upgrade the C/C++ libraries, then you have to refresh Oracle's local copies of these files. This is achieved using the following commands:

```
$ SET DEFAULT ORA_ROOT:[PRECOMP]
$ @BUILD_DIRS
```

6.6 Pro*COBOL Precompiler

Table 6–5 shows the naming conventions for the Pro*COBOL precompiler.

Table 6–5 Pro*COBOL Naming Conventions

Item	Pro*COBOL
Executable	PROCOB.EXE
demo directory	PROCOB2

Pro*COBOL supports static linking with the static client object library, ORA_ROOT:[LIB32]LIBCLNTST10.OLB, as well as dynamic linking with the dynamically loadable client shared library, ORA_ROOT:[LIB32]LIBCLNTSH.SO.

This section discusses the following topics:

- Section 6.6.1, "Pro*COBOL Oracle Runtime System"

6.6.1 Pro*COBOL Oracle Runtime System

The RUN command or a symbol may be used to invoke Pro*COBOL programs. For example:

```
$ RUN SAMPLE1
```

If the program requires input parameters, then define a symbol for the executable and then use the symbol to invoke the program. For example:

```
$ MYPROG  :== $ORA_ROOT:[BIN]MYPROG.EXE
$ MYPROG parameter_list
```

6.6.2 Pro*COBOL Demonstrations

Demonstrations are provided to show the features of the Pro*COBOL precompiler. These programs are located in the ORA_ROOT:[PRECOMP.DEMO.PROCOB2] directory. By default, all programs are linked with the client shared library.

To run the programs, the demonstration tables created by the ORA_ROOT:[SQLPLUS.DEMO]DEMOBLD.SQL script must exist in the SCOTT schema with the password TIGER.

Note: You must unlock the SCOTT account and set the password before creating the demonstrations.

To precompile, compile, and link the SAMPLE1 demonstration program for Pro*COBOL, run the following commands:

Note: The DBA of the site must install Oracle Database Sample Schemas and unlock SCOTT/TIGER. The following is for illustrative purposes only. Contact the DBA of the site for information about the required security settings.

```
$ @ORA_ROOT:[000000]ORAUSER sid
$ sqlplus / as sysdba
SQL> ALTER USER SCOTT ACCOUNT UNLOCK;
SQL> ALTER USER SCOTT IDENTIFIED BY TIGER;
SQL> EXIT
$ SET DEFAULT ORA_ROOT:[PRECOMP.DEMO.PROCOB2]
$ PROCOB INAME=SAMPLE1
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE SAMPLE1.COB
$ LNPROCOB SAMPLE1
$ RUN SAMPLE1
7934
0
```

To create the Pro*COBOL SAMPLE2, 3, 6, 7, 8, 14 demonstrations, run commands similar to SAMPLE1. No data entry is required.

Running the SAMPLE4 demonstration requires the following data entry:

```
$ RUN SAMPLE4
y
7499
0
```

Note: SAMPLE5.PCO is not supported.

Some demonstrations require you to run a SQL script that is located in the ORA_ROOT:[PRECOMP.DEMO.SQL] directory. If you do not run the script, then a message requesting you to run it is displayed. For example, to create the SAMPLE9 demonstration program and run the required ORA_ROOT:[PRECOMP.DEMO.SQL]CALLDEMO.SQL script, enter:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]CALLDEMO.SQL
$ PROCOB INAME=SAMPLE9 SQLCHECK=FULL USERID=SCOTT/TIGER PICX=VARCHAR2
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE SAMPLE9.COB
$ LNPROCOB SAMPLE9
$ RUN SAMPLE9
10
```

Running the SAMPLE10 demonstration requires additional parameters and data entry:

```
$ PROCOB INAME=SAMPLE10 INCLUDE=ORA_ROOT:[PRECOMP.PUBLIC]
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE SAMPLE10.COB
$ LNPROCOB SAMPLE10
$ RUN SAMPLE10
scott
tiger
SELECT empno FROM emp
```

Running the SAMPLE11 demonstration requires additional SQL statements, parameters, and data entry as follows:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]SAMPLE11.SQL
$ PROCOB INAME=SAMPLE11 SQLCHECK=FULL USERID=SCOTT/TIGER PICX=VARCHAR2
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE SAMPLE11.COB
$ LNPROCOB SAMPLE11
$ RUN SAMPLE11
20
```

Running the SAMPLE12 requires additional parameters and data entry as follows:

```
$ PROCOB INAME=SAMPLE12 DYNAMIC=ANSI
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE SAMPLE12.COB
$ LNPROCOB SAMPLE12
$ RUN SAMPLE12
scott
tiger
SELECT * FROM emp WHERE deptno = :b1
30
```

Running the SAMPLE13 demonstration requires additional data entry as follows:

```
$ RUN SAMPLE13
30
20
```

```
10
0
```

Running the LOBDEMO1 demonstration requires additional SQL statements and data entry as follows:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]LOBDEMO1.SQL
$ PROCOB INAME=LOBDEMO1
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE LOBDEMO1.COB
$ LNPROCOB LOBDEMO1
$ RUN LOBDEMO1
1
2
555001212
3
555001212
4
4
111223333
Mickey
1
2
111223333
3
111223333
5
3
111223333
2
5
```

6.6.3 Pro*COBOL User Programs

> **See Also:** The make file for information about creating 32-bit user programs

To create a program, enter commands similar to the following:

```
$ PROCOB INAME=cobfile1
$ PROCOB INAME=cobfil2
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE cobfile1.COB
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE cobfile2.COB
$ LNPROCOB cobfile1 cobfile1,cobfile2
$ RUN cobfile1
```

In this example:

- cobfilen is the COBOL source file for the program
- The first LNPROCOB parameter is the executable program

For example:

To create the myprog program, enter one of the following commands, depending on the source and type of executable that you want to create:

- To link with the client shared library, enter the following commands:

  ```
  $ PROCOB INAME=myprog
  $ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE myprog.COB
  $ LNPROCOB myprog
  ```

- To link with the client object library (non-shared), enter the following commands:

```
$ PROCOB INAME=myprog
$ COB /ANSI /NAMES=AS_IS /FLOAT=IEEE myprog.COB
$ DEFINE ORA_OLB ORA_ROOT:[LIB32],ORA_RDBMS
$ DEFINE ORA_UTIL ORA_OLB
$ LOUTL MYPROG MYPROG MYPROG NS
```

6.6.4 FORMAT Precompiler Option

The FORMAT precompiler option specifies the format of input lines for COBOL. If you specify the default value `ANSI`, then columns 1 to 6 contain an optional sequence number, column 7 indicates comments or continuation lines, paragraph names begin in columns 8 to 11, and statements begin in columns 12 to 72.

If you specify the value `TERMINAL`, then columns 1 to 6 are dropped, making column 7 the left-most column.

6.6.5 Pro*COBOL Restriction

The use of incorrectly aligned binary data, such as COMP-1, in Pro*COBOL applications will generate unaligned access warnings that will prevent optimum compiler performance. However, the application results are not affected.

6.7 Pro*FORTRAN Precompiler

Before you use the Pro*FORTRAN precompiler, verify that the correct version of the compiler is installed.

This section discusses the following topics:

- Section 6.7.1, "Pro*FORTRAN Demonstrations"
- Section 6.7.2, "Pro*FORTRAN User Programs"

6.7.1 Pro*FORTRAN Demonstrations

Demonstrations are provided to show the features of the Pro*FORTRAN precompiler. All of the demonstrations are located in the `ORA_ROOT:[PRECOMP.DEMO.PROFOR]` directory. By default, all programs are dynamically linked with the client shared library.

To run the programs, the demonstration tables created by the `ORA_ROOT:[sqlplus.demo]demobld.sql` script must exist in the `SCOTT` schema with the password `TIGER`.

> **Note:** You must unlock the `SCOTT` account and set the password before creating the demonstrations.

Before creating the demonstrations, review the `DEMO_PROFOR.MK` make file, which is located in the `ORA_ROOT:[PRECOMP.DEMO.PROFOR]` directory.

For example, to precompile, compile, and link the `SAMPLE1` demonstration program, enter the following commands:

> **Note:** The DBA of the site must install Oracle Database Sample Schemas and unlock SCOTT/TIGER.
>
> The following is for illustrative purposes only. Contact the DBA of the site for information about the required security settings.

```
$ @ORA_ROOT:[000000]ORAUSER sid
$ sqlplus / as sysdba
SQL> ALTER USER SCOTT ACCOUNT UNLOCK;
SQL> ALTER USER SCOTT IDENTIFIED BY TIGER;
SQL> EXIT
$ SET DEFAULT ORA_ROOT:[PRECOMP.DEMO.PROFOR]
$ PROFOR INAME=SAMPLE1 INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC])
$ FORT /NAMES=AS_IS /FLOAT=IEEE -
/INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC]) SAMPLE1
$ LNPROFOR SAMPLE1
$ RUN SAMPLE1
7934
0
```

To create the Pro*FORTRAN `SAMPLE2`, `3`, `6`, `7`, and `8` demonstrations programs, run commands similar to those run for `SAMPLE1`. This procedure does not require data entry.

Running the `SAMPLE4` demonstration requires additional data entry as follows:

```
$ RUN SAMPLE4
y
7934
0
```

> **Note:** `SAMPLE5.PFO` is not supported.

Some demonstrations require you to run a SQL script, located in the `ORA_ROOT:[PRECOMP.DEMO.SQL]` directory. If you do not run the script, then a message requesting you to run it is displayed. For example, to create the `SAMPLE9` demonstration and run the required `ORA_ROOT:[PRECOMP.DEMO.SQL]CALLDEMO.SQL` script, run the following commands:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]CALLDEMO.SQL
$ PROFOR INAME=SAMPLE9 INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],
ORA_ROOT:[PRECOMP.PUBLIC]) SQLCHECK=FULL USERID=SCOTT/TIGER
$ FORT /NAMES=AS_IS /FLOAT=IEEE /INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],
ORA_ROOT:[PRECOMP.PUBLIC]) SAMPLE9
$ LNPROFOR SAMPLE9
$ RUN SAMPLE9
10
```

Running the `SAMPLE10` demonstration requires data entry as follows:

```
$ RUN SAMPLE10
scott
tiger
SELECT empno, ename, job FROM emp;
```

Running the `SAMPLE11` demonstration requires additional SQL statements, parameters, and data entry as follows:

```
$ SQLPLUS/NOLOG
SQL> @ORA_ROOT:[PRECOMP.DEMO.SQL]SAMPLE11.SQL
$ PROFOR INAME=SAMPLE11 INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],
ORA_ROOT:[PRECOMP.PUBLIC]) SQLCHECK=FULL USERID=SCOTT/TIGER
$ FORT /NAMES=AS_IS /FLOAT=IEEE /
INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC]) SAMPLE11
$ LNPROFOR SAMPLE11
$ RUN SAMPLE11
20
```

6.7.2 Pro*FORTRAN User Programs

See Also: The make file for information about creating 32-bit user programs

To create a program, enter a commands similar to the following:

```
$ PROFOR INAME=forfile1 INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_
ROOT:[PRECOMP.PUBLIC])
$ PROFOR INAME=forfil2 INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC])
$ FORT /NAMES=AS_IS /FLOAT=IEEE /
INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC]) forfile1.for
$ FORT /NAMES=AS_IS /FLOAT=IEEE /
INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC]) forfile2.for
$ LNPROFOR forfile1 forfile1,forfile2.
$ RUN forfile1
```

In this example:

- `forfile1` and `forfile2` are the FORTRAN source files for the program
- The first `LNPROFOR` parameter is the executable program

For example, to create the `myprog` program from the Pro*FORTRAN source file `myprog.PFO`, enter one of the following commands, depending on the type of executable that you want to create:

- For an executable linked with the client shared library, enter the following commands:

  ```
  $ PROFOR INAME=myprog
  $ FORT /NAMES=AS_IS /FLOAT=IEEE /
  INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC]) myprog.FOR
  $ LNPROFOR myprog
  ```

- For an executable linked with the client object library (non-shared), enter the following commands:

  ```
  $ PROFOR INAME=myprog
  $ FORT /NAMES=AS_IS /FLOAT=IEEE /
  INCLUDE=(ORA_ROOT:[RDBMS.PUBLIC],ORA_ROOT:[PRECOMP.PUBLIC]) myprog.FOR
  $ DEFINE ORA_OLB ORA_ROOT:[LIB32],ORA_RDBMS
  $ DEFINE ORA_UTIL ORA_OLB
  $ LOUTL myprog myprog myprog NS
  ```

6.8 Using the Oracle Call Interface Routines

Oracle Call Interface routines enable 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. The C programming language, FORTRAN, and COBOL are supported on HP OpenVMS for OCI programs.

OCI sample programs are supplied in the following directory:

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

This section discusses the following topics:

- Section 6.8.1, "Guidelines"
- Section 6.8.2, "CDA/LDA Structure Information"
- Section 6.8.3, "Linking Oracle Call Interface Programs Written in the C Programming Language"
- Section 6.8.4, "Linking Oracle Call Interface Programs Written in Other Languages"

6.8.1 Guidelines

The following guidelines apply when using OCI routines:

- You can run OCI programs with the HP OpenVMS debugger by compiling with the `/DEBUG` directive and then linking using the D option of the `LNOCIC` (or `LNOCI`) command file.

  See Also: Section 6.8.3, "Linking Oracle Call Interface Programs Written in the C Programming Language" for information about the syntax of `LNOCIC`

- While using an asynchronous system trap (AST), you are restricted to using only the OBREAK procedure. No other OCI calls can be used.

6.8.2 CDA/LDA Structure Information

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

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

6.8.2.1 Size and Offsets of Structure Elements

The information for the `cda_def` structure, of which `Cda_Def` and `Lda_Def` are type definitions, is shown in Table 6–6.

Table 6–6 lists the sizes and offsets of structure elements.

Table 6–6 Size and Offset of Structure Elements

Structure Element	Offset (Bytes)	Size (Bytes)
v2_rc	0	2
ft	2	2
rpc	4	4
peo	8	2
fc	10	1
rc	12	2
wrn	14	1

Table 6–6 (Cont.) Size and Offset of Structure Elements

Structure Element	Offset (Bytes)	Size (Bytes)
rid	20	16
ose	36	4
rcsp	44	4

6.8.3 Linking Oracle Call Interface Programs Written in the C Programming Language

LNOCIC.COM is used to link Oracle Call Interface routines written in the C programming language. The syntax of this command is as follows:

```
$ LNOCIC executable objfilelist options
```

In this syntax:

- *executable* is the name of the executable image to be created.

  A file name extension is not required.

- *objfilelist* is a list of object files and libraries separated by commas.

  If this list is longer than one line, then use the continuation character, the hyphen (-). Spaces are not allowed in the object file list.

- *options* is a list of options with no separators needed:
  - D: Links with the HP OpenVMS DEBUG utility
  - F: Produces a full map
  - M: Creates a link map
  - X: Produces a link map with cross-references

  For example:

  ```
  $ LNOCIC SAMPLE OBJECT1 D
  ```

6.8.4 Linking Oracle Call Interface Programs Written in Other Languages

LNOCI.COM is used to link with programs that are not written in the C programming language. Of these, only FORTRAN, and COBOL are supported on HP OpenVMS for OCI programs. The syntax is as follows:

```
$ LNOCI executable objectfilelist options
```

For example:

```
$ LNOCI SAMPLE OBJECT1 D
```

Note: The old style Oracle Call Interface (HLI) function calls are not supported in Oracle Database 11*g* Enterprise Edition.

6.9 Data Areas and Data Types

Data types for Oracle on HP OpenVMS are described in the following section. The Cursor Data Area is correct for HP OpenVMS as shown in the programmatic interface guides.

Binary Integers

For HP OpenVMS, binary integers are 32 bits and short binary integers are 16 bits, as shown in Table 6–7.

Table 6–7 lists binary integers for programming languages.

Table 6–7 Usage of Binary and Short Binary Integers

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

6.10 Using Literals as Call Arguments

In FORTRAN, literals and the CHARACTER data type are passed by descriptor to subroutines. Oracle requires all data to be passed by reference. HP OpenVMS FORTRAN provides the `%REF` compiler directive for overriding the normal calling mechanism. This compiler directive should be used to pass literal strings and `CHAR` data to Oracle software routines like OCI.

For example:

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

6.11 Optional or Missing Parameters

On HP OpenVMS, the C programming language does not permit missing optional parameters. All call parameters must be specified. FORTRAN and COBOL, however, allow for missing trailing parameters. The required defaults are automatically provided. FORTRAN also allows missing embedded parameters. The required defaults are automatically provided.

If you omit a parameter by using the -1 convention, then the argument can be either the integer value -1 or a reference to the integer value -1, as long as the argument is of the integer or short binary integer data type. If the argument is the address of any data type, then -1 must be passed by value.

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

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

In COBOL, you could use this:

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

6.12 Using Event Flags

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

Oracle Database 11*g* prevents asynchronous events from interfering with synchronous events by overwriting their event flags. This may increase the reliability of Oracle Database 11*g* software on modern hardware, but it may introduce some problems for application programmers.

In Oracle Database 11*g*, references to event flags 1 through 18 are included in the software. All of these event flags except flags 1 and 5 are tied to specific asynchronous events within Oracle Database 11*g*. Event flags 1 and 5 are used by all synchronous events within Oracle Database 11*g* and can also be used by application programmers. `SYS$GETEF()` is not used for these event flags.

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

6.13 Custom Link Files

Oracle recommends that you use the provided link scripts to create user programs as described in the product-specific sections of this chapter. If you modify the provided link file or if you choose to use a custom-written link file, then the following restrictions apply:

- Do not modify the order of the Oracle libraries. Oracle libraries are included on the link line more than once so that all the symbols are resolved during linking.

  The order of the Oracle libraries is essential for the following reasons:

  – Oracle libraries are mutually referential.

    For example, functions in library A call functions in library B, and functions in library B call functions in library A.

- If you want to add a library to the link line, then add it to the beginning or to the end of the link line. Do not place user libraries between Oracle libraries.
- Oracle library names and the contents of Oracle libraries are subject to change between releases. Always use the link scripts file that ships with the current release as a guide to determine the required libraries.

6.14 Multithreaded Applications

The Oracle libraries provided with this release are thread-safe, which enables support for multithreaded applications.

7

Building and Running Demonstrations

This chapter describes how to build and run the SQL*Loader and PL/SQL demonstrations installed with Oracle Database 11*g*. It contains the following sections:

- Section 7.1, "PL/SQL Demonstrations"
- Section 7.2, "Oracle RDBMS Demonstrations"
- Section 7.3, "Oracle RDBMS C++ File Demonstrations"
- Section 7.4, "Oracle RDBMS Java File Demonstrations"
- Section 7.5, "XDK Demonstrations"
- Section 7.6, "JDBC Demonstrations"
- Section 7.7, "Running Oracle Text and Oracle Spatial Demonstrations"
- Section 7.8, "SQL*Loader Demonstrations"

> **Note:** To use the demonstrations described in this chapter, you must install Oracle Database Examples included on the *Oracle Database 11g Companion CD*.
>
> You must also unlock the SCOTT account and set the password before creating the demonstrations.

7.1 PL/SQL Demonstrations

PL/SQL includes a number of sample programs that you can load. The Oracle Database 11*g* database must be open and mounted to work with the sample programs.

This section contains the following topics:

- Section 7.1.1, "PL/SQL Kernel Demonstrations"
- Section 7.1.2, "PL/SQL Precompiler Demonstrations"

7.1.1 PL/SQL Kernel Demonstrations

The following PL/SQL kernel demonstrations are available:

```
EXAMP1.SQL
EXAMP2.SQL
EXAMP3.SQL
EXAMP4.SQL
EXTPROC.SQL
EXAMP5.SQL
EXAMP6.SQL
```

```
EXAMP7.SQL
EXAMP8.SQL
EXAMP11.SQL
EXAMP12.SQL
EXAMP13.SQL
EXAMP14.SQL
SAMPLE1.SQL
SAMPLE2.SQL
SAMPLE3.SQL
SAMPLE4.SQL
```

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

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

```
$ SET DEFAULT ORA_ROOT:[PLSQL.DEMO]
$ SQLPLUS SCOTT/TIGER
```

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

```
SQL> @EXAMPn.SQL
```

In this command, *n* denotes a unique integer value for each demonstration file.

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

To run the EXTPROC demonstration:

1. Add the following lines to the TNSNAMES.ORA file:

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

2. Add the following line to the LISTENER.ORA file:

```
SID_LIST_LISTENER=(SID_LIST=(SID_DESC= -
(SID_NAME:extproc)(PROGRAM=disk:[<oraclehome>NETWORK.ADMIN]EXTPROC)))
```

3. From SQL*Plus, enter the following commands:

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

4. To run the demonstration, enter the following command:

```
SQL> @extproc
```

7.1.2 PL/SQL Precompiler Demonstrations

The following precompiler demonstrations are available:

```
EXAMP9.pc
EXAMP10.pc
SAMPLE5.pc
```

```
SAMPLE16.pc
```

To build a single demonstration, perform the following steps for the examp9 example.

```
$ SET DEF ORA_ROOT:[PLSQL.DEMO]
$ SQLPLUS SCOTT/TIGER
SQLPLUS>@EXAMPBLD.SQL
SQLPLUS>@EXAMPLOD.SQL
EXIT
$ PROC USERID=SCOTT/TIGER
SQLCHECK=SEMANTICS EXAMP9.PC
$ CC/STANDARD=VAXC EXAMP9.C
$ LNPROC EXAMP9
```

To run the EXAMP9 demonstration, enter the following command:

```
$ RUN EXAMP9
```

7.2 Oracle RDBMS Demonstrations

To build and run the Oracle RDBMS demonstrations:

1. Ensure that the supported version of the C programming language compiler for this release is installed.
2. Set the default directory to ORA_RDBMS_DEMO.
3. Set up the ORA_OLB logical.

   ```
   DEFINE ORA_OLB ORA_RDBMS_DEMO,ORA_RDBMS,ORA_OLB32,ORA_ROOT:[RDBMS.LIB32]
   ```

4. Set up the ORA_UTIL logical.

   ```
   DEFINE ORA_UTIL ORA_OLB
   ```

5. Compile the C programming language file by using the following command:

   ```
   CC/PREFIX=ALL/INCLUD=(SYS$DISK:[],ORA_ROOT:[RDBMS.PUBLIC])/
   FLOAT=IEEE/IEEE_MODE=DENORM/GRAN=BYTE/ARCH=EV56-/
   OPT=TUNE=EV6/EXTE=STRI/PREF=ALL/NOSTAN/NOANS/NAME=(SHORT,AS_IS)DEMO_FILE-+
   DECC$LIBRARY:[000000]DECC$RTLDEF.TLB/LIB
   ```

   In this command, replace *demo_file* with the name of the C programming language file that you want to build.

6. Link the demonstration using LNOCIC:

   ```
   LNOCIC CDEMO1.EXE CDEMO1.OBJ
   ```

   In cases where the command for building the demonstration accepts command-line parameters, you can define a new symbol that is treated as the equivalent of the executable name. For example:

   ```
   $ OCI19 := $ORA_ROOT:[RDBMS.DEMO]OCI19.EXE
   $ OCI19 4
   ```

7.2.1 Extensible Indexing Demonstrations

To run the Extensible Indexing demonstrations:

1. Use the same command for compiling the C file as described in the earlier procedure.

2. For extdemo, extdemo2, extdemo4 and extdemo5, you must create an opt file with the specific entry points for the shared image. The opt file contents for each demonstration are as follows:

   - **EXTDEMO.OPT**

     ```
     case_sensitive=YES
     symbol_vector = (-
     Initialize=PROCEDURE,INITIALIZE/Initialize=PROCEDURE-
     ,Iterate=PROCEDURE,ITERATE/Iterate=PROCEDURE-
     ,Terminate=PROCEDURE,TERMINATE/Terminate=PROCEDURE-
     ,Merge=PROCEDURE,MERGE/Merge=PROCEDURE-
     ,Delete=PROCEDURE,DELETE/Delete=PROCEDURE-
     ,WrapContext=PROCEDURE,WRAPCONTEXT/WrapContext=PROCEDURE-)
     case_sensitive=NO
     ```

   - **EXTDEMO2.OPT**

     ```
     case_sensitive=YES
     symbol_vector = (-
     qxiqtbi=PROCEDURE,QXIQTBI/qxiqtbi=PROCEDURE-
     ,qxiqtbd=PROCEDURE,QXIQTBD/qxiqtbd=PROCEDURE-
     ,qxiqtbu=PROCEDURE,QXIQTBU/qxiqtbu=PROCEDURE-
     ,qxiqtbs=PROCEDURE,QXIQTBS/qxiqtbs=PROCEDURE-
     ,qxiqtbf=PROCEDURE,QXIQTBF/qxiqtbf=PROCEDURE-
     ,qxiqtbc=PROCEDURE,QXIQTBC/qxiqtbc=PROCEDURE-)
     case_sensitive=NO
     ```

   - **EXTDEMO5.OPT**

     ```
     case_sensitive=YES
     symbol_vector = (-
     qxiqtbpi=PROCEDURE,QXIQTBPI/qxiqtbpi=PROCEDURE-
     ,qxiqtbpd=PROCEDURE,QXIQTBPD/qxiqtbpd=PROCEDURE-
     ,qxiqtbpu=PROCEDURE,QXIQTBPU/qxiqtbpu=PROCEDURE-
     ,qxiqtbps=PROCEDURE,QXIQTBPS/qxiqtbps=PROCEDURE-
     ,qxiqtbpf=PROCEDURE,QXIQTBPF/qxiqtbpf=PROCEDURE-
     ,qxiqtbpc=PROCEDURE,QXIQTBPC/qxiqtbpc=PROCEDURE-)
     case_sensitive=NO‘
     ```

3. To build and run EXTDEMO3 demonstrations, the CLASSPATH should be:

   ```
   $define classpath ".:/jdbc_lib/classes12.jar:/sqlj_lib/runtime12.jar:/
   ora_rdbms_
   jlib/xdb.jar:/ora_xdk_lib/
   xmlparserv2.jar:/jdbc_lib/jndi.jar:/jis_lib/jta.jar:/
   ora_rdbms_jlib/ODCI.jar:/ora_rdbms_jlib/CartridgeServices.jar:."
   ```

4. Link the executable with LOUTL using the shared option along with the option file that corresponds to the demonstration you are building. For example:

   ```
   LOUTL EXTDEMO2 EXTDEMO2.OPT/OPT,EXTDEMO2.OBJ EXTDEMO2 I
   ```

5. Create the user for the demonstration and grant the necessary privileges to the user as follows:

   ```
   SQL> connect system/manager
   SQL> drop user extdemo2 cascade;
   SQL> create user extdemo2identified by extdemo2
   default tablespace system quota unlimited on system ;
   SQL> grant connect, resource to extdemo2 ;
   SQL> grant create library to extdemo2 ;
   ```

```
SQL> grant create any directory to extdemo2 ;
SQL> grant drop any directory to extdemo2 ;
SQL> grant create any operator to extdemo2 ;
SQL> grant create indextype to extdemo2 ;
SQL> grant create table to extdemo2 ;
```

6. As mentioned in the step 4 of `extdemo2.sql`, create the associated library in the user schema as follows:

```
connect extdemo2/extdemo2
CREATE OR REPLACE LIBRARY extdemo2l IS
'vqat5:[10ghome.rdbms.demo]extdemo2.exe' ;
```

7. Run the following SQL script:

```
SQL>@extdemo2.sql
```

8. For extended `EXTDEMO4.SQL`, replace the path in the `CREATE LIBRARY` command with the path for the generated executable.

7.3 Oracle RDBMS C++ File Demonstrations

To build and run the Oracle RDBMS C++ file demonstrations:

1. Ensure that the supported version of the C++ programming language compiler for this release is installed.
2. Set the default directory to `ORA_RDBMS_DEMO`.
3. Compile the C++ file by running the following command:

```
$CXX/STANDARD/DEBUG=TRACE/OPTIMIZE/PREFIX=ALL/GRAN=LONG -
/NAMES=(AS_IS,SHORT)-
/INCLUDE=([], ORA_ROOT:[RDBMS.PUBLIC]) -
/NOANSI/EXTERN=STRICT demo_file.CPP
```

   In this command, replace *demo_file*`.CPP` with the name of the file that you want to build.

4. To link the OCCI C++ programming language demonstrations, use the following command:

```
$ LNOCIC demo_file.EXE demo_file.OBJ,ORA_OLB:XAONDY.OBJ,ORA_OLB:LIBOCCI10.OLB/
LIB CPP NS
```

   In this command, replace *demo_file* with the name of the file that you want to build.

7.4 Oracle RDBMS Java File Demonstrations

This section contains the following topics:

- Section 7.4.1, "aqjms Demonstrations"
- Section 7.4.2, "rmanpipe.sql Demonstrations"
- Section 7.4.3, "JavaVM Demonstrations"

7.4.1 aqjms Demonstrations

To build and run the `AQJMS` demonstrations:

1. While performing the steps outlined in the AQJMSREADME.TXT file, make the following changes:

   Note: The AQJMSREADME.TXT can be found under ORA_ROOT:[RDBMS.DEMO].

   a. Replace $ORACLE_HOME in the CLASSPATH specification with the corresponding HP OpenVMS logical name or the abbreviated logical name for the referenced directory. You must set up the JDK version before performing this step. For example:

      Replace $ORACLE_HOME/rdbms/jlib/aqapi13.jar

      with /ORACLE_HOME/rdbms/jlib/aqapi13.jar

      or use /ora_rdbms_jlib/aqapi13.jar

2. In certain cases, the abbreviated form may have to be used to circumvent command line length limitations. Alternatively, the -V option may be used on HP OpenVMS java, and all parameters (including the CLASSPATH) can be placed in a DAT file.

   Note: In the current release, the following demonstrations do not support the oci8 driver on HP OpenVMS:

   - AQJMSDEMO01.JAVA
   - AQJMSDEMO02.JAVA
   - AQJMSDEMO05.JAVA
   - AQJMSDEMO06.JAVA
   - AQJMSDEMO08.JAVA

7.4.2 rmanpipe.sql Demonstrations

The RMANPIPE.SQL script uses VMS_RMAN_PIPE.COM, the HP OpenVMS-specific COM file, to emulate the UNIX operator, which creates a separate, detached process to run the specified command. The COM file dynamically creates a VMSPIPE.COM file. When it is run, the VMSPIPE.COM file creates a corresponding log VMSPIPE.LOG in the ORA_RDBMS_DEMO directory.

7.4.3 JavaVM Demonstrations

The JavaVM demonstrations are available in the following directory:

ORA_ROOT:[JAVAVM.DEMO.EXAMPLES.JSPROC.BASIC]

For each sample, scripts are provided with a name of the form BUILD_RUN_*.COM. When you run these scripts, the corresponding demonstrations are built and run. The output of all demonstrations is directed to SYS$OUTPUT

The NCOMP JavaVM demonstrations are not currently supported on HP OpenVMS.

7.5 XDK Demonstrations

Java and C programming language demonstrations are available for XDK. This section describes how to build and run these demonstrations. It contains the following topics:

7.5.1 Java Demonstrations

To build and run the Java demonstrations, use the scripts provided in the subdirectories of the ORA_ROOT:[XDK.DEMO.JAVA] directory. The names of these scripts are in the form of BUILD_RUN_*.COM. When you run these scripts, the corresponding demonstrations are built and run. You must set the default to the specific demonstration program directory before running any particular script. The result of running the demonstrations are directed to SYS$OUTPUT and are also saved in files that have names of the form *.OUT.

7.5.2 C Programming Language Demonstrations

To build and run the C programming language demonstrations, use the generic scripts provided in the top-level ORA_ROOT:[XDK.DEMO.C] directory:

- COMPILE_SAMPLE.COM

  This script compiles a sample C programming language source file and produces an object file, given the name of the script (without the file name extension) as parameter P1.

- LINK_SAMPLE.COM

  This script links a sample demonstration, given the name of the object file produced by the COMPILE_SAMPLE.COM script as parameter P1.

- BUILD_SAMPLE.COM

  This script combines the actions of the COMPILE_SAMPLE.COM and LINK_SAMPLE.COM scripts.

  To run the demonstration, you must run the executable produced by linking the sample. No arguments are required. The expected output is provided under each sample subdirectory with a file name extension of STD.

7.6 JDBC Demonstrations

The JDBC (DBJava) demonstrations are shipped as a Java.JAR file DEMO.JAR, which are located in the ORA_ROOT:[JDBC.DEMO] directory. To install the demonstrations, enter the following commands:

```
$ SET DEFAULT ORA_ROOT:[JDBC.DEMO]
$ JAR XVF DEMO.JAR
```

1. After installing DEMO.JAR, read the ORA_ROOT:[JDBC.DEMO]SAMPLES-README.TXT file.
2. For any particular example, set the default to the directory where the example resides.
3. Run the corresponding HP OpenVMS DCL command file (VDJDS*.COM) based on the first letters of the directory path, followed by the parameters requested demonstration name (file name without the .JAVA file name extension) and connection method (OCI, OCITNS, or Thin). Modify the command file as required.

   For example, to run the (V)MS (D)b(J)ava (D)emo (S)ample (G)eneric SelectExample using OCI, use the following command:

```
$ SET DEFAULT ORA_ROOT:[JDBC.DEMO.SAMPLES.GENERIC]
$ @ORA_ROOT:[JDBC.DEMO.SAMPLES.GENERIC]VDJDSG.COM SelectExample oci
```

The HP OpenVMS DCL command files `VDJDS*.COM` in each JDBC demonstration directory are analogous to their UNIX Makefile and Microsoft Windows `RUNDEMO.BAT` counterparts.

7.7 Running Oracle Text and Oracle Spatial Demonstrations

The following sections contains information about running the Oracle Text and Oracle Database 11*g* Spatial demonstrations:

- Section 7.7.1, "Oracle Text"
- Section 7.7.2, "Oracle Spatial"
- Section 7.7.3, "Spatial Network Demonstrations"
- Section 7.7.4, "Spatial Example Demonstrations"
- Section 7.7.5, "Spatial Georaster Demonstrations"

7.7.1 Oracle Text

Refer to the `ORA_ROOT:[CTX.SAMPLE.APJ]INDEX.HTML` file and *Oracle Text Reference* for information about the Oracle Text code samples.

7.7.2 Oracle Spatial

Refer to the `ORA_ROOT:[MD.DOC]README.TXT` file for information about the Oracle Database 11*g* Spatial demonstration. Refer to *Oracle Spatial Developer's Guide* for information about Oracle Database 11*g* Spatial.

For the `Spatial Motif` demonstration, refer to `ORA_ROOT:[MD.DEMO.UNIX.MOTIF]README`.

7.7.2.1 Running the Spatial Demonstration

The following is a sample Spatial run:

```
$! On HP OpenVMS at DCL to build SDO Motif demo, run:
$ SQLPLUS/NOLOG
SQL> CONNECT MDSYS/MDSYS
SQL> @ORA_ROOT:[MD.ADMIN]SDOWIN.SQL
SQL> @ORA_ROOT:[MD.ADMIN]PRVTWIN.PLB
SQL> @ORA_ROOT:[MD.DEMO.UNIX.MOTIF.SRC.SQL_SCRIPTS]MY_WINDOW.SQL
SQL> @ORA_ROOT:[MD.DEMO.UNIX.MOTIF.SRC.SQL_SCRIPTS]MY_WIN.SQL
SQL> EXIT
$! OpenVMS Logicals and Symbols already setup for MD_VIEWER, XENVIRONMENT,
motifdemo in:
$ @ORA_ROOT:[MD.PORT.VMS.INSTALL]DEMO_MOTIF.COM ALL
```

1. On a workstation, start an X Window emulator.
2. Find the IP address for the workstation as follows:

```
W2K/Start/Programs/Accessories/Command Prompt
DOS> ipconfig
IP Address 130.35.158.58
DOS> exit

$! Back on OpenVMS enter IP address from above:
```

```
$ SET DISPLAY/CREATE/TRANSPORT=TCPIP/NODE=130.35.158.58
$ RUN SYS$SYSTEM:DECW$CLOCK  ! VERIFY X WINDOW EMULATOR IS RUNNING
$ MOTIFDEMO  ! EXECUTE SDO MOTIF DEMO
Enter username: MDSYS
Enter password: MDSYS
Is database remote [N]: N
```

7.7.3 Spatial Network Demonstrations

Before running `Spatial Network` demonstrations, read `ORA_ROOT:[MD.DEMO.NETWORK...]README.TXT` for each demonstration to be run.

Following are example runs for the `PL/SQL, SQL*Loader Logical`, `SQL*Loader Spatial`, `Java`, and `Network Editor` demonstrations.

SDO Network Example PL/SQL Demonstration

To run the `SDO Network Example PL/SQL` demonstration, run the following at the DCL command prompt:

```
$ SET DEFAULT ORA_ROOT:[MD.DEMO.NETWORK.EXAMPLES.PLSQL]
$ SQLPLUS SCOTT/TIGER @CREATE_LOGICAL.SQL
```

SDO Network Example SQL*Loader Logical Demonstration

To run `SDO Network Example SQL*Loader Logical` demonstration, run the following commands at the DCL command prompt:

```
$ SET DEFAULT ORA_ROOT:[MD.DEMO.NETWORK.EXAMPLES.SQLLDR.LOGICAL]
$ @ORA_ROOT:[MD.PORT.VMS.INSTALL]LOAD_TEST_NET.COM
```

SDO Network Example SQL*Loader Spatial Demonstration

To run `SDO Network Example SQL*Loader Spatial` demonstration on HP OpenVMS, run the following commands at the DCL command prompt:

```
$ SET DEFAULT ORA_ROOT:[MD.DEMO.NETWORK.EXAMPLES.SQLLDR.SPATIAL]
$ @ORA_ROOT:[MD.PORT.VMS.INSTALL]LOAD_SPATIAL_NET.COM
```

SDO Network Example Java Demonstration

To run `SDO Network Example Java` demonstration on HP OpenVMS, run the following commands at the DCL command prompt:

```
$ @ORA_ROOT:[JDBC]JDBC_SETUP_JDK14.COM
SQL> create user MDNETWORK identified by MDNETWORK;
$ SET DEFAULT ORA_ROOT:[MD.DEMO.NETWORK.EXAMPLES.JAVA.DATA]
$ SQLPLUS/NOLOG
SQL> connect / as sysdba
```

The following is for illustrative purposes only. Contact the Security Manager or DBA for information about the correct security settings.

```
SQL> grant all privileges to MDNETWORK with admin option;
SQL> exit
$ SQLPLUS MDNETWORK/MDNETWORK @REMOVE_BI_TEST.SQL
SQL> exit
$ SQLPLUS MDNETWORK/MDNETWORK @REMOVE_UN_TEST.sql
SQL> exit
$ IMP MDNETWORK/MDNETWORK FILE=BI_TEST.DMP TABLES="'BI_TEST_NODE$'"
$ IMP MDNETWORK/MDNETWORK FILE=BI_TEST.DMP TABLES="'BI_TEST_LINK$'"
$ SQLPLUS MDNETWORK/MDNETWORK @BI_TEST_META.SQL
SQL> exit
$ IMP MDNETWORK/MDNETWORK FILE=UN_TEST.DMP TABLES="'UN_TEST_NODE$'"
```

```
$ IMP MDNETWORK/MDNETWORK FILE=UN_TEST.DMP TABLES="'UN_TEST_LINK$'"
$ SQLPLUS MDNETWORK/MDNETWORK @UN_TEST_META.SQL
SQL> exit
$ SET DEFAULT ORA_ROOT:[MD.DEMO.NETWORK.EXAMPLES.JAVA]
$ EDIT ORA_ROOT:[MD.DEMO.NETWORK.EXAMPLES.JAVA]LOADANDANALYZE.JAVA
```

Change `host`, `port`, `sid` using values from the `tnsnames.ora` file.

```
$ ! Enter each java command as all one line
$ JAVAC -CLASSPATH .:'F$TRNLNM("ORACLE_HOME_UNIX")'/JDBC/LIB/
CLASSES12.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/
LIB/XMLPARSERV2.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/
MD/LIB/SDOAPI.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/MD/LIB/SDONM.JAR
LOADANDANALYZE.JAVA
$ JAVA -CLASSPATH .:'F$TRNLNM("ORACLE_HOME_UNIX")'/JDBC/LIB/
CLASSES12.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/
LIB/XMLPARSERV2.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/MD/LIB/
SDOAPI.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/MD/LIB/SDONM.JAR "LOADANDANALYZE"
$ EDIT ORA_ROOT:[MD.DEMO.NETWORK.EXAMPLES.JAVA]CREATEANDSTORE.JAVA
```

Change `host`, `port`, `sid` using values from the `tnsnames.ora` file.

```
$ ! Enter each java command as all one line
$ JAVAC -CLASSPATH .:'F$TRNLNM("ORACLE_HOME_UNIX")'/
JDBC/LIB/CLASSES12.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/LIB/XMLPARSERV2.JAR
:'F$TRNLNM("ORACLE_HOME_UNIX")'/MD/LIB/SDOAPI.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/
MD/LIB/SDORNM.JAR CREATEANDSTORE.JAVA
$ JAVA -CLASSPATH .:'F$TRNLNM("ORACLE_HOME_UNIX")'/JDBC/LIB/
CLASSES12.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/
LIB/XMLPARSERV2.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/
MD/LIB/SDOAPI.JAR:'F$TRNLNM("ORACLE_HOME_UNIX")'/MD/LIB/SDONM.JAR "CREATEANDSTORE"
```

SDO Network Editor Demonstration

On HP OpenVMS at DCL to run `SDO Network Editor` demonstration:

1. Prior to running the `SDO Network Editor` demonstration, set up an X Window emulator.
2. Load the sample data by running commands similar to the following:

```
ORA_ROOT:[MD.DEMO.NETWORK.EXAMPLES.JAVA.DATA] above.
$ SET DEFAULT ORA_ROOT:[MD.DEMO.NETWORK.EDITOR]
$ @ORA_ROOT:[MD.PORT.VMS.INSTALL]STARTNETWORKEDITOR.COM
```

7.7.4 Spatial Example Demonstrations

For information about running the Spatial example demonstrations, read the `ORA_ROOT:[MD.DEMO.EXAMPLES]PARALLEL.DOC` file. The following is an example of a Spatial demonstration:

1. To run `SDO Example Scripts` demonstration, run the following at the at DCL command prompt:

```
$ SET DEFAULT ORA_ROOT:[MD.DEMO.EXAMPLES.SCRIPTS]
$ SQLPLUS /NOLOG
SQL> connect / as sysdba
SQL> create user SDO_USR identified by SDO_USR;
```

The following is for illustrative purpose only:

Note: Contact the site Security Manager or DBA for information about the appropriate security settings.

```
SQL> grant all privileges to SDO_USR with admin option;
SQL> create Tablespace SDO_DATA DATAFILE 'ORA_DB:SDO_DATA.F' SIZE 50M;
SQL> connect sdo_usr/sdo_usr
SQL> @ORA_ROOT:[MD.DEMO.EXAMPLES]PARTITION_POINTS.SQL
```

2. To compile, link, and run `SDO Example` demonstrations, run the following commands at the DCL command prompt:

Note: Before running the `SDO Example` demonstrations, set up an X Window emulator as described in Section 7.7.2.1, "Running the Spatial Demonstration".

```
$ SET DEFAULT ORA_ROOT:[MD.DEMO.EXAMPLES]
$ @ORA_ROOT:[MD.PORT.VMS.INSTALL]DEMO_SDO.COM
$ READGEOM:= $ORA_ROOT:[MD.DEMO.EXAMPLES]READGEOM.EXE
$ READGEOM parameter_list
$ WRITEGEOM := $ORA_ROOT:[MD.DEMO.EXAMPLES]WRITEGEOM.EXE
$ WRITEGEOM parameter_list
$ RUN MIGCTL
$ RUN MIGOCI
```

7.7.5 Spatial Georaster Demonstrations

Before running the `Spatial Georaster` demonstrations, read the `ORA_ROOT:[MD.DEMO.GEORASTER...]README` file for each demonstration to be run. The following is an example of how to run a `Spatial Georaster` demonstration:

1. Run the following commands on HP OpenVMS at DCL to run the `SDO Georaster PL/SQL` demonstrations:

```
$ set default ORA_ROOT:[MD.DEMO.GEORASTER.PLSQL]
$ sqlplus/nolog @GEORASTER_DEMO.SQL
SQL> exit
```

2. After all `Spatial Georaster` demonstrations have been run, enter the following commands to remove the `Georaster PL/SQL` demonstrations:

```
$ sqlplus herman/password @DROP_GEORASTER_TABLE.SQL
SQL> exit
```

To run `SDO Georaster Java` demonstrations, follow the instructions in the `ORA_ROOT:[MD.DEMO.GEORASTER.JAVA]README` file. Before running the `SDO Georaster Java` demonstrations, set up an X Window emulator as described in Section 7.7.2.1, "Running the Spatial Demonstration". The GeoRasterExporter portion of the `GeoRaster` demo is unsupported.

7.8 SQL*Loader Demonstrations

The following sections describe how to build and run the SQL*Loader demonstrations installed with Oracle Database 11*g*.

Review the ULCASE.SH file for an example of how to run all of the SQL*Loader demonstrations. To run an individual demonstration, read the information contained in the file to determine how to run it.

The following SQL*Loader demonstration files are included with Oracle Database 11*g* in the ORA_RDBMS_DEMO directory. Run the demonstrations in numerical order:

```
ULCASE1
ULCASE2
ULCASE3
ULCASE4
ULCASE5
ULCASE6
ULCASE7
```

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

Creating and Running a Demonstration

Note:

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

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

1. Run the ULCASE*n*.SQL script corresponding to the demonstration you want to run:

   ```
   $ SQLPLUS SCOTT/TIGER @ULCASEn.SQL
   ```

2. Load the demonstration data into the database by running the following command:

   ```
   $ SQLLDR SCOTT/TIGER ULCASEn.CTL
   ```

   The following list provides additional information about the ULCASE2, ULCASE6, and ULCASE7 demonstrations:

   - For the ULCASE2 demonstration, you do not have to run the ULCASE2.SQL script.
   - For the ULCASE6 demonstration, run the ULCASE6.SQL script, and run the following command:

     ```
     $ SQLLDR SCOTT/TIGER ULCASE6 DIRECT=TRUE
     ```

   - For the ULCASE7 demonstration, run the ULCASE7S.SQL script, and run the following command:

     ```
     $ SQLLDR SCOTT/TIGER ULCASE7.CTL
     ```

     After running the demonstration, run the ULCASE7E.SQL script to drop the trigger and package used by this demonstration.

7.8.1 Administering SQL*Loader

SQL*Loader is used by both database administrators and Oracle Database 11*g* users. It loads data from standard operating system files into Oracle Database tables.

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

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

Table 7–1 describes the processing options used in the preceding example.

Table 7–1 SQL*Loader Processing Option String

String	Description
`str`	Specifies a stream of records, each terminated by a newline character, which are read in one record at a time
	This string is the default.
`fix`	Indicates that the file consists of fixed-length records, each of which is *n* bytes long, where *n* is an integer.
`var`	Indicates that the file consists of variable-length records, with the length of each record specified in the first *n* characters
	If you do not specify a value for *n*, then SQL*Loader assumes a value of 5.

If you do not select the file processing option, then the information is processed by default as a stream of records (`str`). You may find that `fix` mode gives better performance than the default `str` mode because it does not scan for record terminators.

7.8.1.1 Newline Characters in Fixed-Length Records

When using the `fix` option to read a file containing fixed-length records, where each record is terminated by a newline character, include the length of the newline character (1 character) when specifying the record length to SQL*Loader.

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

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

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

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

7.8.1.2 Removing Newline Characters

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

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

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

8

Tuning Oracle Database 11*g*

This chapter describes how to configure an Oracle Database 11*g* installation to optimize its performance. This chapter contains the following sections:

8.1 Introduction to Tuning

Oracle Database 11*g* is a highly optimizable software product. Frequent tuning optimizes system performance and prevents bottlenecks. .

> **See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* and *Oracle Database Performance Tuning Guide and Reference*

Performance bottlenecks are often caused by the following factors:

- Memory contention

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

- Disk I/O contention

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

- CPU contention

  Although the HP OpenVMS kernel usually allocates CPU resources effectively, many processes compete for CPU cycles, and this can cause contention. If you installed Oracle Database 11*g* in a multiprocessor environment, then there may be a different level of contention on each CPU.

- Oracle resources contention

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

8.2 Oracle Performance Tuning Tools

Various tools are provided for gathering statistics, analyzing performance and tuning performance. Automatic Workload Repository takes snapshots of the system every 60 minutes. Active Session History collects samples of active sessions. New Time Model offers a unique way to store statistics and metrics. Automatic Database Diagnostic Monitor is a powerful self-diagnostic engine that analyzes the system, identifies the major problem in the system, and recommends corrective action. These tools include the `V$` performance views and the `STATSPACK` scripts.

8.3 Oracle SQL Tuning Tools

A variety of tools are provided for tuning SQL, such as SQL Tuning Advisor and SQL Access Advisor. These tools include the `V$` performance views, the `EXPLAIN PLAN` command, the `SQL TRACE` facility, the `TKPROF` facility, the Autotrace report, and the `STATSPACK` scripts.

> **See Also:** *Oracle Database 11g Database Performance Tuning Guide and Reference*

8.4 Tuning Memory Management

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

The Oracle buffer manager ensures that the more frequently accessed data is cached longer. If you monitor the buffer manager and tune the buffer cache, then you can have a significant influence on Oracle Database 11*g* performance. The optimal Oracle Database 11*g* buffer size for the system depends on the overall system load and the relative priority of Oracle over other applications.

This section contains the following topics:

- Section 8.4.1, "Allocate Sufficient Swap Space"
- Section 8.4.2, "Control Paging"
- Section 8.4.3, "Adjust Oracle Block Size"

8.4.1 Allocate Sufficient Swap Space

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

```
$ SHOW MEMORY/FILES
```

If the system is swapping and you must conserve memory, then:

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

See Also:

The operating system documentation for more information about the `$ SHOW MEMORY` command

The HP OpenVMS `INSTALL` utility Help for information about installing additional page and swap files

HP OpenVMS System Manager's Guide for information about managing page and swap files

8.4.2 Control Paging

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

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

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

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

8.4.3 Adjust Oracle Block Size

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

The `DB_BLOCK_SIZE` initialization parameter sets the database block size across the database. To see the current value of the `DB_BLOCK_SIZE` parameter, or to create new tablespaces with a different size, enter the `SHOW PARAMETER` command in SQL*Plus.

8.5 Tuning Disk I/O

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

8.6 Monitoring Disk Performance

To monitor disk performance, use the `$ MONITOR DISK` command.

The average value from `$ MONITOR DISK/ITEM=QUEUE` should not exceed 1.0. If it does or if the MAX value is high, then the system may experience an I/O bottleneck.

8.7 Tuning CPU Usage

Oracle Database is designed to operate with all users and background processes operating at the same priority level. Changing priority levels causes unexpected effects on contention and response times. Oracle Database does not support changing the priority of user and background processes.

For example, if the log writer process (LGWR) gets a low priority, then it is not run frequently enough and LGWR becomes a bottleneck. In contrast, if LGWR has a high priority, then the response time for user processes may be poor.

8.8 System Global Area

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

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

- DB_CACHE_SIZE
- DB_BLOCK_SIZE
- JAVA_POOL_SIZE
- LARGE_POOL_SIZE
- LOG_BUFFERS
- SHARED_POOL_SIZE

Alternatively, set the SGA_TARGET initialization parameter to enable Oracle to automatically tune the SGA size.

Exercise caution when setting values for these parameters. When values are set too high, too much of the physical memory is devoted to shared memory, resulting in poor performance.

Oracle Database installations configured with Shared Server require a higher setting for the SHARED_POOL_SIZE initialization parameter, or a custom configuration that uses the LARGE_POOL_SIZE initialization parameter. If you installed the database with Oracle Universal Installer, then the value of the SHARED_POOL_SIZE parameter is set automatically by Oracle Database Configuration Assistant. However, if you created a database manually, then increase the value of the SHARED_POOL_SIZE parameter in the parameter file by 1 KB for each concurrent user.

8.8.1 Determine the Size of the SGA

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

- Enter the following SQL*Plus command to display the size of the SGA for a running database:

  ```
  SQL> SHOW SGA
  ```

  The result is shown in bytes.

- Determine the size of the SGA when you start the database instance. The SGA size is displayed next to the heading Total System Global Area.

8.9 Enhanced Oracle Performance

Install Oracle image as an HP OpenVMS resident image to increase the performance of Oracle. To install the ORACLE.EXE image resident, the following changes need to be made to HP OpenVMS and Oracle.

Note: Make valid backups of all HP OpenVMS and Oracle software before any changes are implemented.

Oracle recommends you to do initial testing in a non-production environment whenever possible. For queries, contact your local support organization.

8.9.1 System Changes

The following system changes are recommended for improved Oracle performance:

- SYSGEN
- Oracle Command Procedures

SYSGEN

Increase the current size of the SYSGEN parameters: GH_RES_CODE, GH_RES_DATA, and GH_RSRVPGCNT. The size of the parameters GH_RES_CODE and GH_RSRVPGCNT need to be increased by 8192, and GH_RES_DATA to its current maximum value of 2048. This increase needs to accommodate the size of the image being installed resident.

Oracle recommends using AUTOGEN and the MODPARAMS.DAT file for making these changes. These values should be changed by making use of the ADD_*parameter* feature of AUTOGEN.

See Also: Refer to:

- *HP OpenVMS System Services Reference Manual* for more information about SYSGEN and SYSGEN parameters
- *HP OpenVMS System Manager's Manual, Volume 2: Tuning, Monitoring, and Complex Systems* for information about using AUTOGEN to adjust system parameters

Note: These parameters are not dynamic. You must restart your system for the new values to take effect.

You must also install the HP OpenVMS image SYS$LIBRARY:DSMTSHR.EXE with shared address space before the Oracle image is installed resident. HP OpenVMS installs this image by default, but not with shared address space.

There are two ways to implement this:

- Run the INSTALL REPLACE command as shown:

  ```
  $ INSTALL RELACE SYS$LIBRARY:DISMNTSHR.EXE/SHARED=ADDRESS
  ```

  This command would need to be rerun after any system restart and before the Oracle image is installed (which typically occurs during execution of Oracle startup procedures).

- Edit the file SYS$MANAGER:VMS$IMAGES_MASTER.DAT, which contains the list of installed images for HP OpenVMS, and add the /SHARED=ADDRESS qualifier for the DISMNTSHR.EXE entry in the file. This is done before running AUTOGEN to ensure that DISMNTSHR.EXE is installed with the /SHARED=ADDRESS option after a system restart or after running AUTOGEN. You may re-edit this file if it is included in a system

patch or upgrade. Caution should be taken when editing this file, so that this is the only change made.

Oracle Command Procedures

Changes to the command procedure which links the `ORACLE.EXE` image and the procedure which installs that image need to be made for the Oracle image to be installed resident. These changes are shown below:

- For `ORA_ROOT:[000000]INSORACLE.COM`

  Old:

  ```
  INSTALL CREATE ORA_ROOT:[BIN]ORACLE.EXE/OPEN/HEADER_RES/SHARE
  ```

  New:

  ```
  INSTALL CREATE ORA_ROOT:[BIN]ORACLE.EXE/OPEN/HEADER_RES/SHARE=ADDRESS/RESIDENT
  ```

A

Database Limits

This appendix describes the database limits.

A.1 Database Limits

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

> **Note:** Interdependencies between these parameters may affect allowable values.

Table A–1 CREATE CONTROLFILE and CREATE DATABASE Parameters

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

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

Table A–2 File Size Limits

File Type	Maximum SIze
Data files	4,194,303 multiplied by the value of the `DB_BLOCK_SIZE` parameter
Import/Export files and SQL*Loader files	Unlimited
Control files	192000 database blocks

B

Managing the Database

Ensuring that Oracle Database 11*g* operates successfully can involve tuning the system or modifying parameters. These tasks require a thorough understanding of HP OpenVMS system administration as well as the concepts documented in *Oracle Database Administrator's Guide*.

This appendix contains the following topics:

B.1 SQL*Plus and Oracle Net Services

When you start SQL*Plus, a Bequeath protocol adapter connection is made if no TNS connect descriptor is supplied. Refer to Chapter 5, "Configuring Oracle Net Services" for more information about Bequeath adapter.

B.2 Creating Multiple Control Files

Three control files are created whenever you create a database. By default, the files are named `CONTROL01.CTL`, `CONTROL02.CTL`, and `CONTROL03.CTL`. They reside in the directory pointed to by the `ORA_DB` logical name. However, Oracle recommends that you back up the control files and create additional copies. When working with control files, keep in mind the following:

- When you add more control files, you must add the new file names and locations to the `CONTROL_FILES` initialization parameter.
- By default, the control files reside in the `ORA_DB` directory.
- Control files can be moved to any location.
- To guard against device failure, control files should be placed on separate devices.

See Also: *Oracle Database Administrator's Guide*

B.3 Managing Database Files

During the Oracle Database installation procedure, you can create one database file in the directory referenced by the logical name ORA_DB, typically ORA_ROOT:[ORADATA.*dbname*]SYSTEM01.DBF.

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

This section discusses the following topics to manage database files:

- Section B.3.1, "Using Commands to Manage Database Files"
- Section B.3.2, "Adding Files"
- Section B.3.3, "Renaming Files"
- Section B.3.4, "Moving Tablespace Files"
- Section B.3.5, "Moving Redo Log Files"

B.3.1 Using Commands to Manage Database Files

This section discusses commands that you can use to manage database files.

See Also: *Oracle Database Administrator's Guide*

ALTER DATABASE

You can use the ALTER DATABASE command to mount, open, or close a database, to add or drop redo log files, and to archive redo log files. You can also use this command to rename 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.

DROP TABLESPACE

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

B.3.2 Adding Files

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

B.3.3 Renaming Files

If the name of the physical device is somehow dissociated from the database file locations, then the database cannot access these files. Use the ALTER DATABASE command to rename the file to its current location. After renaming the files, shut down the database and then back up the control files as in the following example:

```
SQL> ALTER DATABASE RENAME FILE
2> 'DISK$1:[ORACLE11G.oradata.V10TEST]SYSTEM01.DBF' TO
3> 'MY$DISK:[ORACLE11G.oradata.V10TEST]SYSTEM01.DBF'
SQL> EXIT
$ BACKUP/LOG/VERIFY/-
DISK$1:[ORACLE11G.oradata.V10TEST]*.CTL -
```

```
MY$DISK:[ORACLE11G.oradata.V10TEST]*.CTL
```

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

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

B.3.4 Moving Tablespace Files

To move a tablespace file to a new location:

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

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

2. Back up the tablespace and control files that you want to move.
3. Copy or move the file to a new location.

   Use BACKUP/VERIFY/DELETE to move the file.

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

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

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

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

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

6. Back up the control files.

B.3.5 Moving Redo Log Files

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

1. Identify the fully qualified file name of the redo log files that you want to move by using one of the following methods:

a. Start the database.

b. Run the following query:

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

2. Shut down the database, create a backup 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 copies of all database, control, and redo log files as a precaution against any problems that can arise during this procedure.

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

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

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

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

The file names specified must be correct and the new files must already exist. If either of these requirements is not met, then the statement fails.

4. Shut down the database by using the following command:

```
SQL> SHUTDOWN
```

5. Back up the control files for safety.

6. Restart the database using the following commands.

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

B.4 Database Verification Utility

The database verification utility (DBV) is the preferred technique for verifying the integrity of the database. Run this utility with the DBV symbol. Since Oracle Database

11*g* release 1, `DBV` can be used on an open database.

To verify data in an Oracle Database 11*g* Release 2 (11.2) database, point to the data files from the Oracle Database 11*g* Release 2 (11.2) installation.

> **See Also:** *Oracle Database Utilities* for information about using SQL*Plus to verify the database

B.5 Important Note on Changes to Data File Formats for HP OpenVMS

In Oracle Database 11*g*, the transportable tablespace feature has been extended to enable tablespaces to be transported across different platforms. To make this feature available on HP OpenVMS has necessitated a change in the Oracle file format. Specifically, data files, created while the database is running in 11*g* compatibility mode, are created with a new header block. This is called the OSD header (also referred to as block zero) at the beginning of the file. There are several important points to note regarding support for this new feature on HP OpenVMS:

- Oracle Database 11*g* retains full backward compatibility with 9.2 data files, which on HP OpenVMS prior to 11*g* did not contain a block 0. That is, a database can be started in 11*g* compatibility mode with a mix of 9.2 and 11*g* data files and is fully operable (all updates, writes, read, all operations available on a 11*g* data file are available on the 9.2 data file).
- Data files created while the database is running in the 9.2 compatibility mode will continue to be created without the new header block.
- The cross-platform transportable tablespace feature can only be used on files which have a block 0. For 11*g* data files, this is the case by default. For 9.2 data files, an explicit transformation must be applied to the file to create a new data file, which will contain the required block zero. Oracle Database 11*g* recommends the use of the `RMAN backup as copy datafile` command for the process of creating a new 11*g* format data file, that is:

  ```
  RMAN> backup as copy datafile 'tbs_31.f' format '11g_tbs_31.f';
  ```

  This creates a new data file, which is a copy of the original data file but with the new block 0 header and a 11*g* format generic file header.

  > **See Also:** For more information about the `rman backup` command, refer to the relevant sections in the Oracle documentation set

C

Backing Up and Archiving the Database

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

This chapter contains the following topics:

- Section C.1, "Archiving Redo Log Files"
- Section C.2, "Backing Up the Database"
- Section C.3, "Exporting to and Importing from Multiple Tapes"
- Section C.4, "Recovering Data"

C.1 Archiving Redo Log Files

In case a media failure occurs, the extent of database recovery depends on whether or not you archive the redo logs and how often you back up and export the database.

> **See Also:** *Oracle Database Administrator's Guide* for more information about archiving

Information in the redo logs is always sufficient to guarantee 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, then you must shut down Oracle Database 11*g* before backing up the database.

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

Three initial redo logs of 100 MB each are created during installation. These initial logs are created in `NOARCHIVELOG` mode. You can change them to `ARCHIVELOG` mode with the `ALTER DATABASE` command. You can create additional logs with the `ALTER DATABASE` command. To see the current status of the log files, use the `ARCHIVE LOG LIST` command.

This section contains the following topics:

- Section C.1.1, "Specifying Archive Destinations"
- Section C.1.2, "Archiving Automatically"
- Section C.1.3, "Archiving Manually"

C.1.1 Specifying Archive Destinations

You can archive redo log files to disk. If you want to archive redo logs to tape, then you must first archive them to disk, and then use the HP OpenVMS BACKUP utility to copy them from disk to tape. You should never archive directly to tape.

> **See Also:** Hewlett-Packard's document *HP OpenVMS Guide to Tapes and Devices* and *Oracle Database Administrator's Guide*

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 HP OpenVMS.

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

```
LOG_ARCHIVE_DEST   = DISK$ARC:[ORACLE.V11g.ORADATA.PROD]
LOG_ARCHIVE_FORMAT = MIS_SEQ%s_SCN%c.ARC
```

For faster failure 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 the names of all the redo log files that are archived.

C.1.2 Archiving Automatically

If a database is running with ARCHIVELOG mode enabled, then the redo log files of a given instance must be archived manually or automatically..

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

1. Shut down the current instance.
2. Set the value of the LOG_ARCHIVE_START parameter in the INIT.ORA file to TRUE.
3. Specify the destination of the archived files with the LOG_ARCHIVE_DEST parameter in the same parameter file (either the instance-specific INIT*sid*.ORA file, or the INIT.ORA file itself).
4. Restart the instance.

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

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

The next time an online redo log file must be archived for the current instance, it will be archived automatically until the instance is shut down next. To archive permanently, you must set the LOG_ARCHIVE_START, LOG_ARCHIVE_DEST, and LOG_ARCHIVE_FORMAT parameters in the required parameter file, the INIT.ORA file or the *setup-node_sid*_INIT.ORA parameter file of the instance.

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

C.1.3 Archiving Manually

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

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

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

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

- To archive the next file, use the NEXT option as follows:

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

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

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

When archiving manually, errors are written to the 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.

C.2 Backing Up the Database

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

- Section C.2.1, "Backing Up a Closed Database"
- Section C.2.2, "Backing Up an Open Database"
- Section C.2.3, "Backing Up Data Structures and Definitions"

Both types of backup restore either all or part of the database to the same condition that existed at the time of backup. To recover any transactions committed after the backup, the DBA must use the redo logs where the transactions were recorded. If you back up files while the database is running, then 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 HP OpenVMS BACKUP utility. Follow the instructions in *Oracle Database Administrator's Guide* to back up both open and closed databases. When you are ready to complete the step that instructs you to perform the actual backup, run the HP OpenVMS BACKUP utility.

C.2.1 Backing Up a Closed Database

A backup of a closed database is also known as an offline or cold backup.

To back up a closed database, complete the following:

1. Shut down all instances by using the `SHUTDOWN NORMAL` command.
2. Run the HP 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 the database file is named `SYSTEM01.DBF` and you are copying to a directory named `ARCDIR`, then enter the following command:

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

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

> **Note:** You can automate much of the backup procedure through the use of scripts or the Recovery Manager (RMAN).

C.2.2 Backing Up an Open Database

A backup of an open database is also known as an online or hot backup.

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

> **Caution:** Do not take the tablespace offline or shut down the system until END BACKUP is completed. The backup may not be usable.
>
> If the following warning message is displayed during the backup procedure, then ignore it and continue with the backup:
>
> ```
> %BACKUP-W-ACCONFLICT, is open for write by another user
> ```

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

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

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

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

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

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 the database files do not reside in the ORA_DB directory, then you may need to specify a directory location other than ORA_DB.

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

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

Note: The BEGIN BACKUP and END BACKUP commands are vital. Backups are unusable if these commands are not used in the preceding steps.

Repeat Steps 1 through 3 for all tablespaces that you want to back up.

Note: You can automate much of the backup procedure through the use of scripts or the Recovery Manager (RMAN).

C.2.3 Backing Up Data Structures and Definitions

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

Note: IMPORT/EXPORT messages are directed to SYS$ERROR, not SYS$OUTPUT, and can be saved to a file if you use the LOGFILE option.

You can export the entire database or portions of the database. You can also perform incremental exports, which save only tables that changed since the last export. These exports are quicker and more convenient. To recover the export file generated by the EXPORT utility, use the IMPORT utility.

Note: You can copy export files to tape if you specify a block size of 4096 bytes.

See Also:

- *HP OpenVMS Database Utilities* for information about using the EXPORT and IMPORT utilities
- *Oracle Database Backup and Recovery Reference* for information about Recovery Manager for backup

This section contains the following topics:

- Section C.2.3.2, "Exporting to Non-HP OpenVMS Systems"

C.2.3.1 Exporting to Other HP OpenVMS Systems

To export files to tape for transferring to another HP OpenVMS system, run the following commands:

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

Several prompts are displayed at this point. You must respond to these prompts as required. 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 ends, enter the following commands:

```
$ DISMOUNT tape_device_name
$ DEALLOCATE tape_device_name
```

C.2.3.2 Exporting to Non-HP OpenVMS Systems

To export files to tape for transfer to a non-HP OpenVMS system, run 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 are displayed at this point. You must respond to these prompts as required. 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 ends, run the following commands:

```
$ DISMOUNT tape_device_name
$ DEALLOCATE tape_device_name
```

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

C.3 Exporting to and Importing from Multiple Tapes

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

- Section C.3.1, "Exporting with Multi-Reel Files"
- Section C.3.2, "Exporting to Multiple Tapes"
- Section C.3.3, "Importing from Multiple Tapes"

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

C.3.1 Exporting with Multi-Reel Files

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

C.3.2 Exporting to Multiple Tapes

To export to multiple tapes, run 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 name of the export file as shown 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. Ensure that a tape drive is allocated.

   The tape rewinds and dismounts by itself. A request number and a message instructing you to mount the second tape is displayed.

2. Make a note of the request number.

3. Mount the next tape.

4. Enter the following command:

   ```
   $ REPLY/TO=request_number
   ```

   In this command, replace *request_number* with the request number you noted in Step 2.

Repeat this procedure as many times as required.

C.3.3 Importing from Multiple Tapes

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

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

To import from multiple tapes, run 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 file name as follows:

```
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. Ensure that a tape drive is allocated.

   The tape rewinds and dismounts by itself. A request number and a message instructing you to mount the second tape is displayed.

2. Make a note of the request number.
3. Mount the next tape.
4. Enter the following command:

   ```
   $ REPLY/TO=request_number
   ```

   In this command, replace *request_number* with the request number you noted in the Step 2.

   Caution: Initializing the tape destroys the export.

Repeat this sequence as many times as required.

C.4 Recovering Data

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

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

- Section C.4.1, "Overview of Data Recovery"
- Section C.4.2, "Recovering from Instance Failure"
- Section C.4.3, "Recovering from Media Failure"

C.4.1 Overview of Data Recovery

Recovering an Oracle Database 11*g* database is the process of restoring normal Oracle Database 11*g* operations when they are interrupted by operating system error, hardware failure, or process termination. Recovery procedures should ensure that no transactions are lost and that no data is written incorrectly. Consequently, you must back up the database regularly.

The first step in recovering normal Oracle Database 11*g* operation is to determine the type of failure that has occurred. There are four types of failure, but only two require action:

- Instance failure
- Media failure

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

The other two types of failure, statement failure and process failure, result in automatic recovery.

> **See Also:** *Oracle Database Administrator's Guide* for more information about statement and process failure

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

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

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

> **See Also:** *Oracle Database Administrator's Guide* and *Oracle Database Utilities* for information about the Oracle Database 11*g* utilities used in recovery procedures

C.4.2 Recovering from Instance Failure

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

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

- Both committed and uncommitted transactions recorded in the redo logs are rolled forward.
- Uncommitted transactions are rolled back.
- All locks on Oracle Database 11*g* resources are released.

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

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

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 causes instance failure. If possible, the problem that caused process failure should be diagnosed and corrected to avoid its recurrence.

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

C.4.3 Recovering from Media Failure

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

This section contains the following topics:

C.4.3.1 Media Recovery

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

> **See Also:** *Oracle Database Administrator's Guide*

If media failure occurs, then it is unlikely that any of the instances are still operational.

If you must use an archived redo log file during any of these procedures, then use the HP 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. Wildcard characters are not accepted.

C.4.3.2 Using an Export File for Media Recovery

If you decide to import from an export file as part of media recovery, then you must re-create the database using the SQL*Plus utility before importing the export file. The procedure is as follows:

> **See Also:** *Oracle Database Utilities* for information about using an export file for media recovery

1. Back up the current database, redo log, and control files with the HP OpenVMS `BACKUP` utility.
2. Refer to *Oracle Database Utilities* for further information.

D

Logical Names and Parameters

This appendix provides information about Oracle Database 11*g* logical names and utilities, and the default and recommended values for various initialization parameters. Refer to *Oracle Database Administrator's Guide* for general information about all the initialization parameters.

This appendix contains the following topics:

- Section D.1, "Oracle Database 11g Logical Names"
- Section D.2, "System-Dependent Initialization Parameters"

D.1 Oracle Database 11*g* Logical Names

During installation, several logical names are set up. These assignments are referenced through the `ORA_ROOT:[BIN]ORA_DB_LOGICALS.COM` file that is referenced whenever you start, upgrade, link, or relink Oracle Database 11*g* or other Oracle products.

This section contains the following topics:

- Section D.1.1, "Process Quota Logical Names"
- Section D.1.2, "Logical Name Definitions for the MTS Dispatcher"

D.1.1 Process Quota Logical Names

If you do not set quotas for the background processes, then Oracle Database 11*g* uses built-in formulas to determine how to set the quota logical names.

Table D–1 shows each quota logical name, the minimum and maximum values that you can use if you are setting the logical names, and the current formula.

Table D–1 Components of Quota Logical Names

Calculation Component	Value or Formula
`COMFORT_ZONE`	2.5 MB
`P0_DYNAMIC_SIZE`	Process private storage + room for expansion. 20 MB
`P0_IMAGE_SIZE`	30 MB size of P0 image
`P0_TABLE_SIZE`	Size of page tables needed to map SGA
`PQL$_PGFLQUOTA`	`PAGE_TABLE_SIZE`(SGA)+`P0_DYNAMIC_SIZE`+`COMFORT_ZONE`

Table D–1 (Cont.) Components of Quota Logical Names

Calculation Component	Value or Formula
PQL$_WSEXTENT	If backing file used: PAGE_TABLE_SIZE (SGA) + 4 * P0_IMAGE_SIZE + P0_DYNAMIC_SIZE + COMFORT_ZONE))/512 + 1 Without backing file used: 4 * (P0_IMAGE_SIZE + P0_DYNAMIC_SIZE + - COMFORT_ZONE))/512 + 1
PQL$_WSQUOTA	If backing file used: (SGA_SIZE/512 + PAGE_TABLE_SIZE (SGA_SIZE) + 4 * (P0_IMAGE_SIZE (PAGE_IMAGE_SIZE) + .6 * (P0_DYNAMIC_SIZE) + COMFORT_ZONE))/512 +1 Without backing file used: PAGE_TABLE_SIZE (SGA_SIZE) + 4 * (PAGE_TABLE_SIZE (P0_IMAGE_SIZE) + .6 * (P0_DYNAMIC_SIZE) + COMFORT_ZONE))/512 +1

See Also: Section 5.5.3, "Process Quotas" for more information about modifying Oracle process quotas through logical names

D.1.2 Logical Name Definitions for the MTS Dispatcher

In Oracle Database 11*g*, the MTS Dispatcher requires larger BIOLM and ASTLM process quotas than in past releases.

For each number of anticipated MTS Dispatcher connections, both BIOLM and ASTLM must be set to 100 + 5 x *number_of_simultaneous_connections*. For example, if it is required that the MTS Dispatcher should handle 100 connections, then set both BIOLM and ASTLM to 100 + (5 * *100*) = 600. When determining the maximum number of connections, you must consider both inbound and outbound connections. Outbound connections could be made while establishing database links or links to LDAP servers. Failure to set these quotas results in the MTS Dispatcher becoming blocked in a mutex wait state. Therefore, it is better to estimate the highest number of connections possible.

Both quotas may be controlled by setting the systemwide PQL$_BIOLM and PQL$_ASTLM values. They may also be controlled by adding instance-specific or process-specific logical names to ORA_ROOT:[BIN]ORA_DB_LOGICALS.COM.

The process-specific logicals for these are ORA_*sid*_D*xxx*_PQL$_BIOLM and ORA_*sid*_D*xxx*_PQL$_ASTLM.

The instance-wide logicals for these are ORA_*sid*_PQL$_BIOLM and ORA_*sid*_D*xxx*_PQL$_ASTLM.

Both logical names must be set in the system logical name table. For example, to configure one dispatcher for 100 connections, add the following command to ORA_ROOT:[BIN]ORA_DB_LOGICALS.COM:

```
"$DEFINE/SYSTEM ORA_your_sid_name_D000_PQL$_BIOLM 600
```

Note: For more information about setting the BIOLM and ASTLM values, refer to My Oracle Support Note 156484.1:

https://support.oracle.com/epmos/faces/DocumentDisplay?id=156484.1

D.2 System-Dependent Initialization Parameters

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

This section contains the following topics:

D.2.1 BACKGROUND_ DUMP_DEST

Purpose

Identifies the directory where the trace files created by the detached Oracle Database 11*g* processes are sent.

Recommended Value

None

Default Value

ORA_ROOT:[ADMIN.*dbname*.BDUMP]

Distributed Value

None

D.2.2 DB_BLOCK_SIZE

Purpose

Identifies size (in bytes) of Oracle Database 11*g* database blocks and the database buffers in the SGA.

Recommended Value

A binary multiple of 512 bytes (which is the HP OpenVMS I/O block size). The maximum value is 32768.

Default Value
8192

Distributed Value
None

D.2.3 LOG_ARCHIVE_DEST

Purpose
Specifies a default text string to indicate the location and name of the disk file when archiving log files. Archiving directly to tape is not supported.

Recommended Value
Any valid disk file name.

Default Value
`ORA_ARCHIVE`

Distributed Value
None

D.2.4 LOG_ARCHIVE_ FORMAT

Purpose
Specifies a default file name 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.

See Also: *Oracle Database Administrator's Guide*

Recommended Value
Any valid file name format.

Default Value
`ARCH%T_%S_%R.ARC`

Distributed Value
None

D.2.5 PRE_PAGE_SGA

Purpose
Determines whether or not the SGA pages will be paged into each user's working set at connect time. This parameter can be manipulated to reduce page faults.

Recommended Value
Define this parameter as `TRUE` if the current system load has not produced a high rate of page faults.

Default Value

FALSE

Distributed Value

None

D.2.6 SHARED_POOL_SIZE

Purpose

Determines the size of the shared pool. The shared pool contains shared cursors and stored procedures.

Recommended Value

Larger values of this parameter improve performance in multiuser systems. Smaller values use less memory. This parameter's minimum is 300 KB and its maximum is determined by the size of the SGA. Although there are no SGA size limitations, the minimum value is 30 MB.

Default Value

160 MB

Distributed Value

None

D.2.7 SORT_AREA_SIZE

Purpose

Identifies the size of real memory (in bytes) that will be available for sorting processes

Recommended Value

The amount of real memory that you can reasonably expect to have available for sorting. For example, on a system with 256 MB of real memory, with 1/8 available to sort processes and 4 sorts occurring at the same time, you may set this parameter to 256/8/4 = 8 MB.

Default Value

Generally, a large size improves the efficiency of large sort operations only. In most cases, however, the default works for most database operations.

Distributed Value

None

D.2.8 USER_DUMP_DEST

Purpose

Identifies the location to which trace files created by user processes are sent.

Recommended Value

None

Default Value

ORA_ROOT:[ADMIN.*dbname*.UDUMP]

Distributed Value

None

E

Messages and Codes

This appendix covers Oracle Database 11*g* messages, codes, and actions. The information in this appendix supplements *Oracle Database Error Messages.* Refer to this guide for a complete list of messages and detailed information about Oracle messages and codes.

All messages between 07500 and 07999 are specific to the HP OpenVMS operating system.

%DCL-W-ACTIMAGE: error activating image *image_name*

Cause: This is an HP OpenVMS error message that occurs when you try to run an Oracle Database 11*g* tool without installing the Oracle Database 11*g* client sharable image.

Action: Install Oracle Database 11*g* in shared memory before the instance is started by executing the following command file:

```
$ INSORACLE
```

ORA-01031:insufficient privileges

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

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

Privileges to use the `CONNECT /` command depend on whether or not:

- An `ORA_`*`sid`*`_DBA` identifier is in the HP OpenVMS rights database
- The account has the process rights identifier `ORA_DBA`, `ORA_`*`sid`*`_DBA`, or both

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

- If the identifier, `ORA_`*`sid_x`*`_DBA`, exists in the HP OpenVMS rights database for instance *`sid_x`*, then your account must have been granted the process rights identifier `ORA_`*`sid_x`*`_DBA` to control instance *`sid_x`*.
- If the identifier `ORA_`*`sid_x`*`_DBA` exists in the HP OpenVMS 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`*. However, it may control all other instances that do not have `ORA_`*`sid_x`*`_DBA` identifiers defined for them.

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

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

Cause: File is not created because allowing DBAs to perform file operations if the User Identification Code (UIC) group of this account is less than or equal to the SYSGEN parameter MAXSYSGROUP poses a security risk.

Action: Ensure that the DBA creating or opening database files, redo log files, and so on, has a UIC group greater than MAXSYSGROUP.

ORA-07516:sfccf: $open file error

Cause: HP OpenVMS system service $OPEN failed.

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

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

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

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

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

Cause: File is not created because allowing Oracle Database 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 MAXSYSGROUP SYSGEN parameter. If any valid Oracle files exist with such ownership conditions, then you must change their ownership before attempting to run the REUSE command on them.

ORA-07520:sfccf: illegal logical block size

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

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

ORA-07521:sfccf: $create file error

Cause: HP OpenVMS system service $CREATE failed.

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

ORA-07522:sfccf: new file exists

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

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

ORA-07526:sfifi: illegal logical block size

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

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

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

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

Action: Ensure that the DBA creating or opening database files, redo log files, and so on, has a UIC group greater than MAXSYSGROUP.

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

Cause: File is not created because allowing Oracle Database 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, then you must change their ownership before attempting to open them.

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

Cause: File is not created because allowing Oracle Database 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, then you must change their ownership before attempting to REUSE them. Likewise, if you attempt to create a file that will inherit an invalid ownership from the parent directory, then you should create it in a different location, or take other steps to avoid this situation.

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

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

Action: Examine the system error, and correct the file name syntax.

ORA-07546:sfcmf: new file exists

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

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

ORA-07547:sfcmf: $OPEN failure

Cause: HP OpenVMS system service $OPEN failed.

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

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

Cause: Too many test files open.

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

ORA-07553:sfofi: out of open files

Cause: The number of open files has exceeded an HP OpenVMS Oracle Database compile time limit.

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

ORA-07556:sfotf: $create error

Cause: HP OpenVMS system service `$CREATE` failed.

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

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

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

Action: This is an internal error. Contact customer support.

ORA-07558:ssfctf: $create error

Cause: HP OpenVMS system service `$CREATE` failed

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

ORA-07560:sltln: $trnlog error

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

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

ORA-07563:sldext: $PARSE failure

Cause: HP OpenVMS system service `$PARSE` failed.

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

ORA-07564:sldext: wildcard in filename or extension

Cause: A wildcard was used in the file name.

Action: Reenter the file name completely.

ORA-07565:sldext: $SEARCH failure

Cause: HP OpenVMS system service `$SEARCH` failed.

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

ORA-07568:slspool: $OPEN failure

Cause: HP OpenVMS system service `$OPEN` failed.

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

ORA-07572:szrfc: insufficient rolename buffer space

Cause: An OS role name was too long.

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

ORA-07573:slkhst: could not perform host operation

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

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

ORA-07582:spstp: SID has illegal value

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

Action: Refer to the *Oracle Database Administrator's Guide* for information about setting the *SID*.

ORA-07585:spdcr: $PARSE failure

Cause: HP OpenVMS system service $PARSE failed.

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

ORA-07586:spdcr: $SEARCH failure

Cause: HP OpenVMS system service $SEARCH failed.

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

ORA-07587:spdcr: $CREPRC failure

Cause: HP OpenVMS system service $CREPRC failed.

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

ORA-07620:smscre: illegal database block size

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

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

ORA-07622:smscre: $CREATE failure

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

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

ORA-07623:smscre: $CRMPSC failure

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

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

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

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

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

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

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

If the available global pages are fragmented, then restart the system after increasing the SYSGEN parameter, GBLPAGES (global page limit). This parameter cannot be dynamically increased. you must restart the system for these changes to take effect. If the available global pages are merely fragmented, but their number is sufficient and restarting the system is enough. In such a case, there is no need to increase the SYSGEN parameter, GBLPAGES.

ORA-07625:smsget: $MGBLSC failure

Cause: While mapping the system global area (SGA) during logon, the HP OpenVMS system service $MGBLSC failed. The usual reason is that Oracle Database 11*g* has not been started up.

Action: Examine the system error message, and refer to the HP OpenVMS system documentation. Start Oracle Database 11*g* 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 the program and try again. Report this error to Oracle Support Services.

ORA-07627:smsfre: $CRETVA failure

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

Action: Examine the system error message, and refer to the HP 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 the program and try again, and report this to Oracle Support Services.

ORA-07636:smsdbp: $MGBLSC failure

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

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

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

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

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

ORA-07647:sszfck: $OPEN failure

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

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

ORA-07655:slsprom: $TRNLOG failure

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

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

ORA-07688:smscre: $CREATE_REGION_64 failure

Cause: HP OpenVMS system service `$CREATE_REGION_64` failed.

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

ORA-07689:smscre: $CRMPSC_GFILE_64 failure

Cause: HP OpenVMS system service `$CRMPSC_GFILE_64` failed.

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

ORA-07690:smscre: $CRMPSC_GDZRO_64 failure

Cause: HP OpenVMS system service `$CRMPSC_GDZRO_64` failed.

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

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

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

Action: Add `ORA_SGA` identifier to the system.

ORA-07692:ssmsget: $MGBSLC_64 failure

Cause: HP OpenVMS system service `$MGBLSC_64` failed.

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

ORA-07693:ssmsget: $DELTVA_64 failure

Cause: HP OpenVMS system service `$DELTVA_64` failed.

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

ORA-07694:ssmsget: $CREATE_REGION_64 failure

Cause: HP OpenVMS system service `$CREATE_REGION_64` failed.

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

ORA-07696:smsfre: $DELETE_REGION_64 failure

Cause: HP OpenVMS system service `$DELETE_REGION_64` failed.

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

ORA-07697:smscre: $GRANTID failure

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

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

ORA-07698:smsget: $GRANTID failure

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

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

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

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

Action: Specify a valid archival device.

ORA-07741:slemop: $OPEN failure

Cause: HP OpenVMS system service `$OPEN` failed.

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

ORA-07822:sspscm: SYS$CREMBX failure

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

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

F

Process Control

This appendix presents some useful tips about managing HP OpenVMS processes. It contains the following topics:

- Section F.1, "Interrupting and Terminating Oracle Operations"
- Section F.2, "Running Oracle Programs as Detached Processes"

> **See Also:** The operating system documentation for more information about these topics

F.1 Interrupting and Terminating Oracle Operations

The following sections explain how to cancel an operation without terminating, how to cancel an operation with the option to continue, and how to disable the control keys.

This section contains the following topics:

- Section F.1.1, "Canceling Without Terminating the Oracle Image"
- Section F.1.2, "Canceling with the Option to Continue"
- Section F.1.3, "Disabling Control Keys"

F.1.1 Canceling Without Terminating the Oracle Image

To cancel an operation without terminating the Oracle image, press Ctrl+C. The current query is canceled. After pressing Ctrl+C, you may need to press Enter to display the prompt again. You may have to do this when you use a command line tool, such as SQL*Plus.

F.1.2 Canceling with the Option to Continue

You can also terminate any Oracle operation by pressing Ctrl+Y. This returns the DCL prompt ($) with a message that Oracle has been interrupted.

To continue the Oracle session, enter `CONTINUE`. To terminate the session, type `EXIT`.

Entering `EXIT` or any other HP OpenVMS command cancels the query and shuts down the tool. Entering `EXIT` causes a noticeable delay before the DCL prompt returns or before the requested HP OpenVMS command runs because a partial shutdown of the Oracle session occurs.

F.1.3 Disabling Control Keys

To disable the control keys, enter the following command:

```
$ SET TERM/PASTHRU
```

F.2 Running Oracle Programs as Detached Processes

Sometimes, you may want to run programs as detached processes. For example, you may want to run a Pro*C program while you are logged into SQL*Plus or while doing work unrelated to working with Oracle Database.

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 stops because it cannot find the logical names referenced by the program.

You can work around this problem by starting the login process `LOGINOUT`, which maps DCL into the virtual space of the detached process. This can run 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. Start the program to be run. For example:

   ```
   $ RUN/DETACH/INPUT=TEST.COM/OUTPUT=TEST.LOG SYS$SYSTEM:LOGINOUT
   ```

   In this command, `TEST.COM` is:

   ```
   $ @DISK$TEST:[ORACLE11G]ORAUSER SID
   $ RUN myprog.EXE
   ```

You may need to include certain process quotas to map DCL into the detached process's virtual space.

> **See Also:** The operating system documentation for more information

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X