Oracle® Database

Database Installation and Administration Guide

19c for Fujitsu BS2000

F28348-02
January 2021

Oracle Database Database Installation and Administration Guide, 19c for Fujitsu BS2000

F28348-02

Primary Author: Binika Kumar

Contributing Authors: Bharathi Jayathirtha, Walter Moser, Wolfgang Schmidt, Helmut Tronberend, Prakash Jashnani

Contents

Part II Installation and Database Creation

2 Oracle Database Installation and Deinstallation

Preface

This manual provides information about installing, administering, maintenance, and usage of Oracle Database and related products. It also provides information for BS2000 end users of Oracle products. Certain topics in this manual apply to both users and database administrators. This manual describes the following:

- How Oracle Database operates under BS2000
- How to install Oracle Database
- How to create or upgrade an Oracle Database
- How to administer an Oracle Database
- How to use Oracle Database utilities
- How to build and run Oracle Database applications

This section contains the following topics:

Audience

This manual is intended for:

- Database Administrators (DBAs)
- Users of Oracle Database
- Oracle Database Support

The reader is assumed to have a fundamental knowledge of BS2000. No attempt is made to document features of BS2000, except as they affect or are affected by Oracle Database.

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.

Using Oracle Database Documentation

Oracle Database products that run on BS2000 principally offer the same functionality as Oracle Database products on other operating systems. However, because of the diversity of operating systems, the use of applications may differ slightly between different operating systems. As a result of this, Oracle provides two types of documentation:

Type	Meaning/Usage
Generic	This is the primary Oracle Database documentation, which describes how the product works and how it is used. Use this type of documentation to learn about product functions and how to use any Oracle Database product or utility.
System Specific	This documentation provides the information required to use the product on a specific operating system. Use this type of documentation to determine whether there are any system-specific deviations from the generic documentation.

This manual is written for users of Oracle Database for BS2000, providing them with BS2000-specific information about using Oracle Database products. It does not describe how to use a product unless its use is different than that is described in the generic documentation.

Related Documents

For more information, refer to the Fujitsu documentation at:

https://bs2manuals.ts.fujitsu.com

All Oracle documentation is available at the following URL:

http://docs.oracle.com/en/

Conventions Used in this Manual

The following conventions are used in this manual:

Typographic Conventions

The following text conventions are used in this manual:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.

Convention	Meaning
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.

Command Syntax

Item	Syntax
Commands	This font identifies text that must be entered exactly as shown: `set echo off`
Variables	Variables appear in italics. Substitute an appropriate value, for example: *arg1*
Required Items	Required items are enclosed in braces { }. You must choose one of the alternatives. `DEFINE { macro1 | macro2 }`
Optional Items	Optional items are enclosed in square brackets []. [*options*] *formname* [*userid/password*]
Repetitive Items	An ellipsis, ... represents an arbitrary number of similar items. CHKVAL *fieldname value1 value2... valueN*

Punctuation

The following symbols should always be entered as they appear in the command format:

Name	Symbol
ampersand	&
backslash	\
colon	:
comma	,
double quotation mark	"
equal sign	=
hyphen	-
number sign	#
parentheses	()
period	.
semicolon	;
single quotation mark	'

Part I
Concepts

This part provides the basic concepts and architecture of Oracle Database on Fujitsu BS2000 systems. It contains the following chapter:

- Concepts and Architecture

1
Concepts and Architecture

The platform independent concepts and the architecture of Oracle Database are described in *Oracle Database Concepts*.

This chapter describes the basic concepts of Oracle Database specific to Fujitsu BS2000 systems. It includes the following topics:

1.1 About the BS2000 Operating System

Oracle Database 19c for Fujitsu BS000 runs on the BS2000 operating system. Oracle Database uses both native BS2000 interface and the POSIX interface.

The sections in this chapter describe in detail how Oracle Database uses the program environments, the native BS2000 and POSIX.

1.2 About File Systems

The default file system for an Oracle Database program depends on the program environment in which the program is started.

Oracle Database 19c operates on the following file systems:

- BS2000 Data Management System (DMS), which is a flat file system.
- POSIX file system, which is hierarchically structured and consists of directories and files.

If you start an Oracle Database utility in the BS2000 program environment, then a flat file is located in the BS2000 DMS. If you start the same Oracle Database utility in the POSIX shell, then a flat file is located in the POSIX file system.

A file system can be directly addressed independent of the program environment. The prefix `/BS2/` addresses the BS2000 DMS file system. A file name that starts with a valid POSIX path, such as `/home/oracle` addresses the POSIX file system.

As in earlier releases, database files always reside in the BS2000 DMS file system. For example, data files, control files, online redo log files, and archive redo log files.

Starting with Oracle Database 12*c* Release 1, most of the Oracle supplied SQL scripts are installed in the POSIX file system. To run a SQL script from the `ORACLE_HOME` directory in the BS2000 program environment, you must enter the fully qualified file name.

Automatic Diagnostic Repository (ADR) requires a hierarchical file system to create a file-based repository for database diagnostic data. So, the directories and files of ADR reside in the POSIX file system.

During Oracle Database installation, executables, libraries, and other files are installed in both the BS2000 DMS and in the POSIX file system. You must provide access to the POSIX file system.

> **See Also:**
>
> "Oracle Database Installation and Deinstallation" for information about access to the POSIX file system

1.3 About Processes

The concepts of Oracle Database processes are not BS2000-specific.

All processes of an Oracle Database instance, for example, server processes and background processes, run as BS2000 tasks.

Oracle Database utilities that are started in the BS2000 program environment are executed in the BS2000 login task. For example, SQL*Plus.

If you start an Oracle Database utility in the POSIX shell, then a new POSIX process is created.

Client processes running in the POSIX shell connect to an instance in the same way as clients in the BS2000 environment. The dedicated server process starts in the BS2000 program environment as a BS2000 task.

Related Topics

- *Oracle Database Concepts*

1.4 Memory Architecture

The concepts of Oracle Database basic memory architecture are not BS2000-specific.

Oracle Database defines different data areas in the main memory, such as SGA and PGA. On BS2000 systems, the SGA is realized as a shared memory pool to which all background and server processes of an instance have read/write access. The PGA is a local memory that is specific to an Oracle background process or a server process.

Most of the Oracle code binaries are placed in code areas in the main memory, which reside in shared memory pools in the BS2000 user address space. These data and code areas must reside at the same virtual addresses in all the server and background tasks. Typically, the default values provided with Oracle Database are sufficient. Address space planning, that is, explicit placement of Oracle Database code or data areas may be required in some special situations when you encounter address

space conflicts. For example, dynamic subsystems may occupy the default address ranges, which may require you to relocate Oracle Database areas.

The following Oracle environment variables control explicit placement of Oracle Database code and data areas:

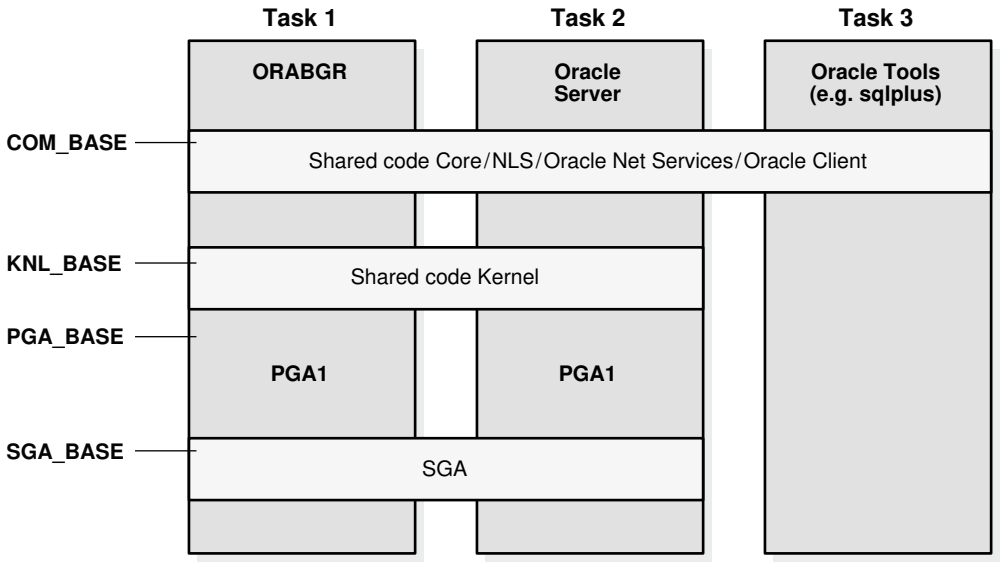
- COM_BASE
- KNL_BASE
- PGA_BASE
- SGA_BASE

The order of the areas in the address space is not significant. The *xxx*_BASE variable is evaluated only during startup processing.

The first process which realizes that a shared pool does not exist creates this shared pool. All other processes of an Oracle Database instance and server processes attach to the existing shared memory.

The following figure is an example of the placement of data areas:

Figure 1-1 Placement of Data Areas in Background, Server and Utility Tasks



The *xxx*_BASE values must be compatible with the BS2000 value SYSBASE (defined by BS2000 generation and delimiting the user address space).

Database application programs use a separate shared code pool for common services such as Core, Globalization Support, and Net Services. The name of this pool is `Client Common Pool` and its placement can be controlled by the ORAENV environment variable CLN_BASE. The default value for the CLN_BASE is set to 200 MB.

```
CLN_BASE=200MB
```

In general, Oracle Database administrators must be aware of conflicts between Oracle pool placements and other pool placements in the system.

Related Topics

- *Oracle Database Concepts*

1.5 BS2000 User Ids

The following BS2000 user IDs are required for Oracle Database:

- Installation User ID
- DBA User ID
- Client User IDs

1.5.1 Installation User ID

The installation user ID is required for installing Oracle Database. The POSIX user, which corresponds to the BS2000 installation user ID must be initialized with a unique user number, group number, and a valid home directory.

A separate installation user ID is required for every Oracle Database release. However, multiple databases using the same release must refer to the same installation user ID.

Oracle Database software uses `$ORACINST` as a placeholder for the installation user ID. During installation, `ORACINST` is replaced by the name of the installation user ID. In this guide, `$ORACINST` is used as a proxy for the installation user ID.

The installation procedure installs Oracle Database in the BS2000 file system (DMS) and in the POSIX file system. The BS2000 file system (DMS) contains:

- Executable programs, such as SQL*Plus.
- Load libraries, in particular, `ORALOAD.LIB`, from which modules are loaded for execution.
- Files and system configuration files specifying default precompiler options.
- Object libraries required to link precompiler applications, such as `PRO.LIB`.
- Platform-specific installation utilities, for example BS2000 command procedures.
- Default configuration files, such as the default `ORAENV` file.
- Files for demo schemas and demo applications.

The POSIX file system contains:

- Oracle supplied SQL scripts.
- Binaries to run Oracle programs, such as SQL*Plus, in the POSIX shell.
- Shared objects, such as `libclntsh.so`.
- JAVA tools like `loadjava` or `owm`.
- Files for application development.

> **Note:**
>
> Starting from Oracle Database 12*c* Release 1, most of the Oracle supplied SQL scripts reside in the POSIX file system. In earlier Oracle Database releases these scripts resided in the BS2000 DMS file system.

1.5.2 DBA User ID

The DBA user ID is used as the owner of one or more Oracle databases. This user ID owns all the files for a specific Oracle database. The corresponding POSIX user must be initialized with a unique user number, group number, and a valid home directory.

All BS2000 tasks of an Oracle Database instance are executed under the DBA user ID. These tasks refer to the executable programs and libraries, which are available under the installation user ID. These programs and libraries must not be copied into the DBA user ID. You can use the installation user ID as a DBA user ID. However, Oracle strongly recommends that you use separate user IDs.

Multiple databases can be created under the same or different DBA user IDs. If the databases are created under different BS2000 user IDs, then the databases are separated and protected from each other by the BS2000 protection mechanisms.

1.5.3 Client User IDs

The Client user ID allows a normal non-DBA user to access and use the database through Oracle Database utilities, such as SQL*Plus, and through application programs.

These client programs can run in the BS2000 user IDs, which are different from the DBA user ID. They can connect to Oracle Database through Oracle Net Services facilities.

The corresponding POSIX user must be initialized with a unique user number, group number, and a valid home directory.

1.6 Oracle Database Programs

Oracle Database programs are stored in program libraries, known as BS2000 LMS libraries.

The programs are loaded from these libraries for execution. They require a specific program environment, which must be defined before execution.

The following topics are discussed:

- Program Libraries
- Program Environment

1.6.1 Program Libraries

Program libraries are required to run Oracle Database programs.

Oracle Database for Fujitsu BS2000 requires the following program libraries:

The ORALOAD Library

All Oracle Database 19c programs require the `ORALOAD` library, which is `$ORACINST.ORALOAD.LIB` by default. Oracle Database uses this library to load executables and subroutines dynamically. The BS2000 link name `ORALOAD` must identify the `ORALOAD` library before starting any Oracle Database program. If the link name is missing, then you get an error message from the BS2000 loader. This link name is set when running the `ORAENV` procedure in the BS2000 program environment.

BS2000 file links are not visible in the POSIX program environment. Oracle Database programs running in the POSIX shell use an internal mechanism to locate the `ORALOAD` library.

The ORAMESG library

The `ORAMESG` library, `$ORACINST.ORAMESG.LIB`, is required for dynamically loading the binaries for the Oracle Database messages by an Oracle Database program when required. The BS2000 link name `ORAMESG` must identify the `ORAMESG` library before starting any Oracle Database program. If the link name is missing, then you get an error message from the BS2000 loader. This link name is set when running the `ORAENV` procedure.

BS2000 file links are not visible in the POSIX program environment. Oracle Database programs running in the POSIX shell use an internal mechanism to locate the `ORAMESG` library.

1.6.2 Program Environment

The program environment for Oracle Database can either be the native BS2000 environment or the POSIX Shell.

The default program environment for server and background tasks is the BS2000 environment. Regardless of the program environment, Oracle Database always requires a running POSIX subsystem.

This section contains the following topics:

1.6.2.1 Oracle Environment Variables

All Oracle Database programs and database applications require environment variables.

Oracle Environment Variables contains a list of Oracle Database environment variables that the database administrator can use. The non-DBA users must set a few of these variables. Any DBA-specific variables that are placed in a non-DBA user's environment are ignored.

1.6.2.2 Setting Variables in the BS2000 Program Environment

Oracle Database utilities and application programs running in the BS2000 program environment use the Oracle Database environment definition file, ORAENV for setting up the program environment.

This file is divided into two parts, an executable part to set required BS2000 file links, and a static part for the definition of the environment variables. During program initialization, the environment variables are read from the ORAENV file.

The procedure $ORACINST.INSTALL.P.DBA automatically creates a template for the ORAENV file with the name *sid*.P.ORAENV, where *sid* is the instance identifier. The generated file provides a default configuration for an Oracle Database instance. You can edit this file to adapt it to your needs. Non-DBA users can also generate an ORAENV file specific to their own environment.

To make the environment variables accessible, run a CALL-PROCEDURE command on the ORAENV file for the instance that you want to use. For example, to specify the environment variables for the instance DEMO, run the following BS2000 SDF command:

```
/CALL-PROCEDURE DEMO.P.ORAENV
```

> **Note:**
>
> - The database administrator must not modify the ORAENV file while an Oracle Database is running.
> - Non-DBA users may modify their ORAENV file at any time.

You can run several Oracle Database instances simultaneously on your BS2000 system, even within the same DBA user ID. A unique system identifier provides a globally unique name for the instance so that a user can select a particular instance by assigning the instance identifier *sid* to the ORACLE_SID environment variable. This is achieved by calling the corresponding ORAENV file *sid*.P.ORAENV.

Alternatively the structured BS2000 SDF-P system variable SYSPOSIX can be used to define the Oracle Database environment variables. The variable SYSPOSIX can be declared in the SYS.SDF.LOGON.USERINCL file so that it is accessible for all programs and procedures started in the BS2000 login task. Use the following command to declare the variable:

```
/DECLARE-VARIABLE
SYSPOSIX,TYPE=*STRUCTURE(DEFINITION=*DYNAMIC),SCOPE=*TASK(STATE=*ANY)
```

If you want to set an environment variable with an underscore in its name, then you must substitute the underscore with a hyphen. For example, to set ORACLE_HOME using the SYSPOSIX system variable, use the following command:

```
/SET-VARIABLE SYSPOSIX.ORACLE-HOME='/u01/app/oracle/product/19.3.0/
dbhome_1'
```

> **See Also:**
>
> - *C Library Functions for POSIX Applications* for more information about the BS2000 SDF-P system variable SYSPOSIX
> - "Generating The Environment-Definition File" for more information

1.6.2.3 Setting Variables in the POSIX Program Environment

Oracle Database utilities and application programs running in the POSIX shell get the environment variables from the POSIX environment.

All Oracle Database and BS2000 specific variables can be set in the POSIX environment. The Oracle Database environment variable ORACLE_HOME must be set. To run a utility for a particular instance, you must also set the Oracle Database environment variable ORACLE_SID. The operating system environment variable PATH must be extended by the path to the Oracle binaries $ORACLE_HOME/bin. If you do not set any other Oracle variable in the POSIX program environment, then Oracle Database utilities will use the default values.

The installation procedure creates a profile in the *oracle_home_path* directory which can be executed to set and export the most important environment variables like ORACLE_HOME, PATH , or LD_LIBRARY_PATH. Use the '.' (dot) command to execute this profile in the current POSIX shell:

```
$ . oracle_home_path/.profile.oracle
```

> **Note:**
>
> The Oracle Database environment variable BGJPAR is marked as a comment after the first installation. So, this variable will not be set when you execute the .profile.oracle command file. If you do not set this variable, then the following default value is used:
>
> ```
> BGJPAR=START=IMME,CPU-LIMIT=NO,LOGGING=*NO
> ```

You can change the default value of a variable by setting this variable in the POSIX environment. For example:

```
$ NLS_LANG=German_Germany.WE8BS2000
$ export NLS_LANG
```

You can also use the ORAENV file that you created in the BS2000 file system DMS. Create a file with the name oraenv*sid* in the $ORACLE_HOME/dbs directory describing the location and name of the ORAENV file.

For example, to use the ORAENV file, $ORADATA.ORCL.P.ORAENV, you must create a file with the name oraenvORCL in the $ORACLE_HOME/dbs directory that contains the name of the BS2000 ORAENV file as follows:

```
$ ORACLE_HOME=/u01/app/oracle/product/19.3.0/dbhome_1
$ export ORACLE_HOME
$ echo '$ORADATA.ORCL.P.ORAENV' > $ORACLE_HOME/dbs/oraenvORCL
```

Set the Oracle Database environment variable ORACLE_SID and call the utility:

```
$ ORACLE_SID=ORCL
$ export ORACLE_SID
$ $ORACLE_HOME/bin/sqlplus /nolog
```

Oracle Database utilities and applications running in the POSIX shell handle the variables of the BS2000 ORAENV file as lower-ranking variables. If a variable is set in the POSIX environment and the same variable is defined in the ORAENV file, then the POSIX variable is not overwritten by the ORAENV variable. For example, if the variable, NLS_LANG is set to German_Germany.WE8BS2000 in the POSIX environment and to American_America.WE8BS2000 in the BS2000 ORAENV file, then the variable keeps the value German_Germany.WE8BS2000 in the environment of the Oracle Database utility running in the POSIX shell.

Consider the following:

- You must set the ORACLE_HOME variable.
- You must set the ORACLE_SID variable to specify a particular instance.
- To access a BS2000 ORAENV file, you must create a file, oraenv*sid* in the $ORACLE_HOME/dbs directory that contains the fully qualified name of your BS2000 ORAENV file.
- The SID in the file name oraenv*sid* is case sensitive and must match the SID specified in the Oracle Database environment variable ORACLE_SID.
- If an ORAENV file in the BS2000 file system is assigned to the specified ORACLE_SID, then ensure that this file is accessible for the utility.
- In the POSIX environment, the variables of the BS2000 ORAENV file are handled as subordinate variables.

1.7 Physical Storage Structures

This section describes some features of physical storage structures, which are specific for Fujitsu BS2000. It includes the following topics:

- Files of an Oracle Database
- Oracle Managed Files
- File of a Bigfile Tablespace

Related Topics

- *Oracle Database Concepts*

1.7.1 Files of an Oracle Database

An Oracle database is a set of operating system files that store data in persistent disk storage.

There are different types of database files:

- data files
- temp files
- control files
- online redo log files
- archive redo log files

The database files always reside in the BS2000 DMS and have the file attribute, `FILE-STRUC=PAM`. Database files cannot reside in the POSIX file system.

Both the BS2000 operating system and Oracle Database perform input and output efficiently in units called blocks. A block is the basic unit of data storage. An Oracle Database block can have the following size depending on the format of the BS2000 DMS pubset:

- 2K, 4K, 6K, 8K, 16K, 32K when using BS2000 2K pubset format
- 4K, 8K, 16K, 32K when using BS2000 4K pubset format

1.7.2 Oracle Managed Files

Oracle Managed Files (OMF) is a file naming strategy that enables you to specify operations in terms of database objects rather than file names.

For example, you can create a tablespace without specifying the names of its data files. In this way, Oracle Managed Files eliminates the need for administrators to directly manage the operating system files in a database.

The following is a list of `INIT.ORA` parameters for Oracle Managed Files:

- `DB_CREATE_FILE_DEST` for data files, temp files, and block change tracking files.
- `DB_CREATE_ONLINE_LOG_DEST_n` for redo log files and control files.
- `DB_RECOVERY_FILE_DEST` for backups, archive log files, and flashback log files.

On BS2000, these parameters are used as a prefix for file names.

Oracle tablespace names can be up to 30 characters long. To associate an OMF created file name with its tablespace, you must use tablespace names that are distinct in the first eight characters. Oracle allows underscore (_) in tablespace names. Any underscore (_) that is present is changed to hyphen (-) for use in the generated file name.

File names for Oracle Managed Files have the following format on BS2000:

File type	Format
control files	destOMC.tttttttt
log files	destOMLlll.tttttttt

File type	Format
permanent tablespace files, data file copies	destOMD.tsn.tttttttt
temporary tablespace files	destOMT.tsn.tttttttt
archive log files	destOMA.Tnnn.Snnnnnn.tttttttt
data file or archivelog backup piece	destOMB.Lnnn.tttttttt
rman autobackup piece	destOMX.xnnnnnnn.tttttttt
block change tracking files	destOMR.tttttttt
flashback log files	destOMF.tttttttt

where:

- *dest* is the destination string (_DEST) in the OMF parameter
- *tttttttt* is the encoded timestamp (which looks like a random mix of letters and numerals)
- *lll* is a three-digit log-group number
- *tsn* is up to eight characters for the tablespace name
- *Tnnn* is the letter "T" followed by a three-digit thread number
- *Snnnnnn* is the letter "S" followed by a six-digit sequence number
- *Lnnn* is the letter "L" followed by a three-digit incremental level
- *x* is the letter *P*, if the database has an `SPFILE`, or the letter *T* if the database does not have an `SPFILE`
- *nnnnnnn* is a seven-byte timestamp

The maximum length of a BS2000 DMS file name is 54 characters. As a consequence, the preceding OMF file name formats impose a limit of 39 characters on `DB_CREATE_ONLINE_LOG_DEST_n` and `DB_CREATE_FILE_DEST`, and 29 characters on `DB_RECOVERY_FILE_DEST`. This limit includes the BS2000 DMS `catid` and `userid` which can have a maximum length of 16 characters in total.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for more information about file name formats

1.7.3 Files of a Bigfile Tablespace

A bigfile tablespace contains a single large data file or a temp file.

This single data file or temp file must reside on a BS2000 DMS pubset with the following attributes:

`LARGE_VOLUMES=*ALLOWED` and `LARGE_FILES=*ALLOWED`

> **See Also:**
>
> *Files and Volumes Larger than 32 GB* for more information about handling large objects on BS2000

1.8 Parameter Files

The database parameter file `INIT.ORA` is used to define startup parameters for the database and the instance.

By default, this file resides in the BS2000 DMS in the DBA user ID. It is also possible to place this file in the POSIX file system.

In addition to the `INIT.ORA` file, a binary server-side initialization file `SPFILE` can be created by using the `CREATE SPFILE` statement. This file must be created in the BS2000 DMS in the DBA user ID.

Related Topics

- *Oracle Database Reference*
- *Oracle Database Administrator's Guide*

Part II
Installation and Database Creation

This part of the document discusses the following topics:

- Oracle Database Installation and Deinstallation
- About Creating a Database
- About Upgrading a Database
- About Upgrading Applications

2

Oracle Database Installation and Deinstallation

This chapter details the preinstallation requirements, postinstallation tasks, installation and deinstallation of Oracle Database 19c on BS2000.

2.1 Overview of Oracle Database Installation

You can download Oracle Database software from Oracle Software Delivery Cloud at `https://edelivery.oracle.com`. The installation files are included in a Library Maintenance System (LMS) library called staging library, which is delivered in a zip file.

Download the zip file and extract it to a temporary location on a Windows or UNIX system. Then upload the staging library to the BS2000 system using File Transfer Protocol (FTP). The following sections describe the installation process for Oracle Database on the Fujitsu BS2000 Servers. The name of the staging library reflects the releases of Oracle Database or Oracle Database Patch Set, and the supported hardware architecture:

- `ora`*`xxxxx`*`.`*`arch`*`.lib`

  Where *`xxxxx`* is the release number and *`arch`* is the supported hardware architecture. For example:

  `ora19300.s390.lib`

> **Note:**
>
> `ora19300.s390.lib` is the name of the staging library for the base release of Oracle Database 19c.

2.2 Planning the Installation

The Oracle Database installation process consists of the following steps:

1. **Read the release notes**: Refer to *Oracle Database Release Notes for Fujitsu BS2000* before you start the installation.
2. **Review the licensing information**: Although the installation media in the media pack contain many Oracle components, you are permitted to use only those components for which you have purchased licenses. Oracle Support Services does not provide support for components for which licenses have not been purchased.

   > **See Also:**
   >
   > *Oracle Database Licensing Information User Manual* for more licensing information

3. **Complete preinstallation tasks**: "Oracle Database Preinstallation Requirements" describes preinstallation tasks that you must complete before installing the product.
4. **Install the software**: "Installing Oracle Database" describes how to use the installation procedure for BS2000 to install Oracle Database 19c.
5. **Complete postinstallation tasks**: "Oracle Database Postinstallation Tasks" describes recommended and required postinstallation tasks.
6. **Multiple software installations**: "About Installing Multiple Oracle Databases" provides information about multiple installations of Oracle software.
7. **Remove Oracle Database software**: "About Removing Oracle Database Software" describes how to remove Oracle Database software from your system.

2.3 Oracle Database Preinstallation Requirements

You must complete the following tasks before you install Oracle Database 19c for Fujitsu BS2000:

- Checking Hardware Requirements
- Checking Software Requirements
- About Checking Required Subsystems
- Checking Network Setup
- Creating Required Operating System Users and Groups
- Required Directories in POSIX
- Identifying or Creating Oracle Base Directory in POSIX

2.3.1 Checking Hardware Requirements

The system must meet the following hardware requirements:

2.3.1.1 Fujitsu BS2000 Server Lines

Oracle Database 19c for Fujitsu BS2000 is based on the /390 instruction set and supports the Fujitsu BS2000 SE Servers.

This release runs on all Fujitsu BS2000 processors, in particular on the /390-based system units of SE servers. It runs in the /390 compatibility mode on processors that are based on Intel x86 architecture. This applies for the x86-based server units of the SE servers (SU x86).

2.3.1.2 Memory Requirements

Oracle Database requires at least 2 GB main memory.

The address space limit for the installation user ID should be 512 MB or higher. For the DBA user IDs, it should be at least 1024 MB.

- To determine the memory size, enter the following command:

  ```
  /SHOW-SYSTEM-INFORMATION INFORMATION=*MEMORY-SIZE
  ```

- To determine the user address space limit, log in to the desired user ID and enter the following command:

  ```
  /SHOW-USER-ATTRIBUTES
  ```

If the user address space is less than the required size, then ask your BS2000 system administrator to increase the address space limit by using the command:

```
/MODIFY-USER-ATTRIBUTES USER-IDENTIFICATION=user-id,ADDRESS-SPACE-
LIMIT={512|1024}
```

2.3.1.3 Disk Space Requirements

Oracle Database 19c requires an installation in the BS2000 file system (DMS) and in the POSIX file system. Oracle recommends using a separate POSIX file system for the Oracle Database software installation.

> **See Also:**
>
> Fujitsu guide for BS2000 mainframes *POSIX Basics for Users and System Administrators* for more information about creating an additional POSIX file system.

When the POSIX administrator or BS2000 system administrator creates a new POSIX file system by using the POSIX installer, which can be started with the `START-POSIX-`

`INSTALLATION` command, a new POSIX container file similar to a UNIX disk partition is allocated in the BS2000 file system. The POSIX file system is created in this container file. If the desired mount point, for example, `/u01`, is not available in the root system, then the mount point is created and the file system is mounted.

Oracle Database 19c software for Fujitsu BS2000 requires the following disk space:

- BS2000 file system:

Item	Free Disk Space
Software	1.300.000 PAM pages
Staging library	1.300.000 PAM pages
Recommended (in total)	3.000.000 PAM pages

- POSIX file system:

Requirement	Free Disk Space
Minimum	4 GB
Recommended	7 GB

> **Note:**
>
> The diagnostic data of the Automatic Diagnostic Repository (ADR) reside in the POSIX file system. The disk space required for the ADR depends on the number of Oracle database instances and the amount of diagnostic data. Therefore, you may need significantly more disk space in the POSIX file system than recommended.

The extracted files occupy about 1.300.000 PAM pages in the BS2000 file system. The files installed in the POSIX file system will occupy about 770 MB.

The staging library does not have to reside in your installation user ID.

2.3.1.4 Display Requirements

Oracle Database supports 9750 compatible terminals. In addition, Oracle Database supports X-terminals in the POSIX environment when you log in to POSIX using `rlogin` or `ssh`.

The minimum resolution for Java based tools with a graphical user interface (GUI) shipped with Oracle Database 19c is 1024 x 768 or higher.

2.3.2 Checking Software Requirements

Before installing Oracle Database, Oracle recommends that you check if your system meets the following software requirements:

- Operating System and Communication System Requirements
- POSIX Parameters

- Package Requirements
- Additional BS2000 Software Components
- Compiler and CRTE Requirements for Oracle Database Applications
- Additional Software Requirements

2.3.2.1 Operating System and Communication System Requirements

Oracle Database 19c requires the following operating system and communication system versions or higher:

- BS2000 OSD/BC V10.0, BS2000 OSD/BC V11.0, and higher.
- *open*Net Server V3.6

  This includes BCAM V23.0 and Sockets V2.7.
- At least the correction package 1/2020 must be installed.

Oracle recommends that you regularly install the most current correction packages for your BS2000 system.

You can check the version of your BS2000 system with the following command:

```
/SHOW-SYSTEM-INFORMATION INFORMATION=*BS2000-ID
```

Note:

- The correction level of the POSIX subsystem must be A45 or higher. You can check the POSIX correction level with the following command:

  ```
  /EXECUTE-POSIX-CMD CMD='uname -v'
  ```

- You can check the version of your *open*Net Server software by checking the version of the software component BCAM with the following command:

  ```
  /SHOW-INSTALLATION-PATH INSTALLATION-UNIT=BCAM,LOGICAL-
  IDENTIFIER=*NONE
  ```

- You can also ask your BS2000 system administrator to check the BCAM version with the command:

  ```
  /BCSHOW SHOW=BCAM
  ```

2.3.2.2 POSIX Parameters

Check the following POSIX parameters for the recommended values for Oracle Database 19c:

Parameter Name	Description	Recommended Value
HDSTNI	Number of hard disk server tasks	4
NPBUF	Number of physical I/O buffers	20
NPROC	Maximum number of processes	1000
NBUF	Number of I/O buffers	200
MAXUP	Maximum number of processes per user	300
NOTTY	Maximum number of ttys	512
NOPTY	Maximum number of ptys	512
NOSTTY	Maximum number of sttys	1024
DBLSTATE	Initial state of POSIX loader	1
DBLPOOL	Size of pool in MB for POSIX loader	30

The POSIX administrator or the BS2000 system administrator can check the values in the file `$TSOS.SYSSSI.POSIX-BC.version`, for example, version = 100|110. The values can also be checked by using the `usp` POSIX command :

```
# usp -s nproc
```

The POSIX administrator or the BS2000 administrator can update the value with the following command:

```
# usp -P parameter -v value
```

> **Note:**
>
> - If you edit the parameters in the `$TSOS.SYSSSI.POSIX-BC.version` file, then you must restart the POSIX subsystem.
> - The parameter values depend on the overall load of the BS2000 system and must be adjusted to this load. The values listed previously are minimal recommendations.
> - Irrespective of the installed software component, that is, a client or a server, Oracle Database requires a running POSIX subsystem.

2.3.2.3 Package Requirements

Oracle Database 19c requires the following POSIX packages with the specified versions or higher:

- POSIX-SH V10.0
- JENV V8.1 or JENV V9.0
- APACHE V2.4A
- PERL V52.4A

- BCAM V23.0

Use the BS2000 command, `SHOW-INSTALLATION-PATH` to check the correction level of the APACHE and Perl products. For example:

```
/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=APACHE,LOGICAL-IDENTIFIER=*NONE
```

Use the following POSIX command to check the installed packages:

```
$ pkginfo package-name
```

> **Note:**
>
> - Ensure that all the specified packages are installed before installing Oracle Database.
> - If you don't have the required APACHE and PERL releases, please request a correction delivery from the Fujitsu software distribution center or contact Fujitsu support.

2.3.2.4 Additional BS2000 Software Components

Oracle Database 19c requires the following additional software with the specified versions or higher:

- CRTE-BASYS V10.0B04 or CRTE-BASYS V11.0B04
- SDF V4.7
- SDF-P V2.5
- SDF-A V4.1
- EDT V17.0D
- LMS V3.5B
- openSM2 V19.0
- PTHREADS V1.4

Note:

- Ensure that the correction package 1/2020 or higher is installed.
- If you don't have the required release of CRTE-BASYS, then please request a correction delivery from the Fujitsu software distribution center or contact Fujitsu support.
- The subsystem `CRTEBASY` must be loaded.
- SDF-A is optional. SDF-A is only required to update the SDF user syntax file for Oracle. If SDF-A is not available, then the SDF user syntax file update for Oracle is skipped. If you do not update the SDF syntax file for Oracle, then you cannot start Oracle Database utilities with BS2000 SDF commands in UNIX-style.
- The software component `PTHREADS` is required by the replay client utility, `wrc`.

2.3.2.5 Compiler and CRTE Requirements for Oracle Database Applications

If high-level languages, such as C or COBOL, are used to interface with the Oracle Database, then the following versions or higher are supported:

- COBOL85 V2.3
- COBOL2000 V1.5
- CPP V3.2
- CRTE V10.0B, V11.0B, and higher

2.3.2.6 Additional Software Requirements

openUTM V6.5 or higher is required if you want to use Oracle Database in configurations with the transaction monitor openUTM.

2.3.3 About Checking Required Subsystems

For a proper behavior, Oracle Database requires running subsytems. It is mandatory to create the following subsytems:

- POSIX
- SOC6
- PRNGD
- CRTEBASY

Use the following BS2000 command to check the state of the subsystems:

```
SHOW-SUBSYSTEM-STATUS
```

For example:

```
/SHOW-SUBSYSTEM-STATUS POSIX
```

2.3.4 Checking Network Setup

Typically, the computer on which you want to install Oracle Database is connected to the network as a member of a network domain. Besides the BCAM host name, the host must have a fully qualified name (*hostname.domain*) that can be resolved by a DNS. You must perform the following important checks on a BS2000 computer:

- About Checking BCAM Timer
- About Checking LWRESD
- About Checking Loopback Address
- About Checking the Configuration Files in the POSIX File System

2.3.4.1 About Checking BCAM Timer

Oracle recommends to check the BCAM connection and letter timer. Your BS2000 system administrator can display the timer values with the following command:

```
/SHOW-BCAM-TIMER[TIMER=*STD]
```

The value for the connection timer, also called `CONN` in the output of the STD section must be set to a minimum of 600 seconds if you use an Oracle database server. The value for the letter timer, also known as `LETT` in the STD section must be set to a large value. The value `0` in the output indicates an infinite letter time.

2.3.4.2 About Checking LWRESD

Oracle recommends to use the Light Weight Resolver (LWRESD) for host name resolution operations. You require BS2000 system administrator privilege for checking and administering LWRESD. Ask your BS2000 system administrator to check if LWRESD is running, by using the following command:

```
/SHOW-LWRESD-PARAMETERS
```

For example, the following output shows the parameter file in use:

```
RESOLV-FILE : :PVS1:$TSOS.SYSDAT.LWRESD.012.RESOLV.CONF
```

This parameter file must contain valid name server IP-addresses and a domain or a search list of domain names.

2.3.4.3 About Checking Loopback Address

Check if the `$TSOS.SYSDAT.BCAM.ETC.HOSTS` file contains an entry for the loopback address. For example:

```
127.0.0.1 localhost loopback # local address
```

2.3.4.4 About Checking the Configuration Files in the POSIX File System

Check the network configuration files, `/etc/hosts` and `/etc/resolv.conf`. These files must be identical to the corresponding files in the BS2000 administrator user ID TSOS. If the files are not identical, then consult your BS2000 system administrator to synchronize the files.

2.3.5 Creating Required Operating System Users and Groups

If you are installing Oracle software on your computer for the first time, then you may have to create several BS2000 user IDs, POSIX users, and groups. The BS2000 user ID for the Oracle Database 19c software is called the installation user ID. The user ID where an Oracle database is stored is called the Oracle Database Administration (DBA) user ID.

The name of the POSIX groups can be:

- `oracle`
- `dba`

On BS2000, you can use only one group for both Oracle Database installation and administration. You must assign the same group to both POSIX users, that is, the installation user and the DBA user. This is because programs running in the BS2000 program environment, such as, Oracle background and server tasks, can only be member of the primary group defined in the POSIX user attributes.

This section includes the following topics:

- About Creating the BS2000 Installation User ID
- About Creating the POSIX Group
- About Initializing the POSIX User
- About Creating Users and Groups for Oracle Databases

2.3.5.1 About Creating the BS2000 Installation User ID

The BS2000 system administrator must create an user ID under which you want the Oracle Database 19c software to reside. This user ID is called the Oracle installation user ID or Oracle software owner. Throughout this guide, `ORACINST` is used as a proxy name for the real installation user ID. This user ID does not require any special BS2000 privileges:

- You must not use the BS2000 system administrator user ID TSOS as an Oracle Database installation user ID.
- The user address space limit must be set to at least 512 MB.
- During installation, all files are created with the following file attributes:

  ```
  USER-ACC=ALL-USERS,ACCESS=READ
  ```

- You do not need to define write access for any file after installation.

When the BS2000 system administrator creates the installation user ID with the following command, the corresponding POSIX user is also created:

```
/ADD-USER ORACINST,…
```

The POSIX user attributes, namely, user number, group number, login directory and program, are assigned default values.

2.3.5.2 About Creating the POSIX Group

For installation under POSIX, the POSIX administrator or the BS2000 system administrator must create groups in POSIX. Oracle recommends to use the group, `oracle` for the Oracle software owner, and Oracle database administrator. However, if you want to use the `dba` group for the Oracle database administrator, then you must use this group for the Oracle software owner too.

- If this is not the first Oracle Database installation under POSIX, then you can determine the group name by using the following POSIX command:

  ```
  $ more /var/opt/oracle/oraInst.loc
  ```

  If the `oraInst.loc` file exists, then the output of this command is similar to the following:

  ```
  inventory_loc=/u01/app/oraInventory
  inst_group=oracle
  ```

  The `inst_group` parameter shows the group name of your former Oracle Database installation. In the above example, it is `oracle`.

- To determine if this group is defined with a unique group ID, enter the following command:

  ```
  $ grep 'oracle' /etc/group
  ```

  If the output shows the group name and the group ID is greater than 100, then the group exists.

  If the group ID is 100, then you must change the ID to a distinct value greater than 100, for example, 104.

  If the group cannot be found in `/etc/group`, then ask your POSIX administrator or BS2000 system administrator to add the group name and a unique group number to the `/etc/group` file. The administrator must use a text editor, such as `edt` or `vi` to add a new line with the following specifications:

  `groupname``::`*`groupnumber`*`:`*`user-id`*`[,`*`user-id`*`,...]`

  The following example shows a line in the `/etc/group` file for the `oracle` group:

  ```
  oracle::104:ORACINST,ORACDBA
  ```

  Add further installation user IDs and all DBA user IDs to the line with the Oracle group.

2.3.5.3 About Initializing the POSIX User

Before using the installation user ID as a POSIX user, the POSIX administrator or the BS2000 system administrator must initialize the POSIX user with the following command:

```
/ADD-POSIX-USER USER-NAME=installation-user-id,USER-NUMBER=nnn,GROUP-
NUMBER=oracle-group-number,HOME-DIRECTORY=path[,RLOGIN-ACCOUNT=account]
```

Consider the following requirements:

- The USER-NAME is the installation user ID.
- The USER-NUMBER must be unique and greater than 100.
- The GROUP-NUMBER must be the number of the Oracle installation group and greater than 100.
- The HOME-DIRECTORY must be a valid path. Do not use a path which resides in the root or var file system. Oracle strictly recommends to create a separate file system for the POSIX users. An example for a valid HOME-DIRECTORY is /home/oracinst where /home is the mount point of a POSIX file system.
- The RLOGIN-ACCOUNT can be specified if you allow the remote access by using rlogin or ssh.

2.3.5.4 About Creating Users and Groups for Oracle Databases

An Oracle database in BS2000 resides in separate BS2000 user IDs. This user ID is called the DBA user ID and is different from the Oracle Database installation user ID. The BS2000 system administrator must ensure that the DBA user ID has the following attributes and privileges:

- Set the user's address space limit to at least the following value:

  ```
  ADDRESS-SPACE-LIMIT=1024
  ```

- Set the following parameters for ACCOUNT-ATTRIBUTES:

  ```
  MAX-ALLOWED-CATEGORY=TP
  NO-CPU-LIMIT=YES
  START-IMMEDIATE=YES
  ```

> **Note:**
>
> You must not use the BS2000 system administrator user ID TSOS as an Oracle database DBA user ID under any circumstances.

Oracle recommends that the BS2000 system administrator defines a separate job class for the background and server tasks.

This job class must have the following characteristics:

```
TP-ALLOWED=YES
NO-CPU-LIMIT=YES
JOB-TYPE=BATCH
START=IMMEDIATELY
```

The POSIX user for the DBA user ID must be initialized with a unique user number and group number. The group number must be the same as used for the Oracle Database software owner.

If you create a new DBA user ID for a new Oracle database, then you must also initialize the POSIX user in the same way as described in "About Initializing the POSIX User." If you want to upgrade an existing Oracle database, then you must check if the POSIX user is initialized for the DBA user ID using the following BS2000 command:

```
/SHOW-POSIX-USER-ATTRIBUTES
```

If the output shows default values and the directory is `/home/gast`, then initialize the POSIX user using the following BS2000 command:

```
/ADD-POSIX-USER USER-NAME=dba-user-id,USER-NUMBER=nnn,GROUP-NUMBER=oracle-group-number,HOME-DIRECTORY=path[,RLOGIN-ACCOUNT=account]
```

You can modify the POSIX user attributes of an initialized POSIX user using the following BS2000 command:

```
/MODIFY-POSIX-USER-ATTRIBUTES USER-ID=dba-user-id,USER-NUMBER=nnn,GROUP-NUMBER=oracle-group-number,DIRECTORY=path
```

Your POSIX administrator or BS2000 system administrator must add the new DBA user ID to the group of the Oracle Database software owner in the `/etc/group` file.

The following example shows the entries for the `oracle` and `dba` groups in the `etc/group` file:

```
oracle::504:ORAINST,ORADATA
```

> **Note:**
>
> - Ensure that the home directory does not reside in the `root` or `var` file system.
> - The `USER-NUMBER` must be unique and greater than 100.
> - The `GROUP-NUMBER` must be the same as the Oracle software owner.

2.3.6 Required Directories in POSIX

The following sections describe the directories that you must identify or create in the POSIX file system for Oracle Database:

- About Oracle Base Directory
- About Oracle Inventory Directory
- About Oracle Home Directory

2.3.6.1 About Oracle Base Directory

The Oracle base directory is a top-level directory for Oracle software installations. Oracle recommends to use only one Oracle base directory for all software installations with a path similar to the following:

/mount_point/app/*oracbase*

In this example:

- *mount_point* is the mount point directory for the file system that contains the Oracle software. The examples in this guide use `/u01` for the mount point directory.

- *oracbase* is used as an example for a significant subdirectory name to show that this is the Oracle base for the software installations.
- If you want to use different Oracle base directories, then you can use the operating system user name of the Oracle software owner as the last subdirectory name.

The components */mount_point/app* represent the operating system part of the path. Before you install Oracle Database 19c, you must create the operating system part of the path.

> **Note:**
>
> - The installation user ID and the installation group must have read, write, and execute permissions to the subdirectory `app`.
> - During Oracle Database software installation, you are prompted for a valid Oracle base directory. You can accept the default value or enter a new path as specified above. The installation procedure creates the Oracle base directory as a subdirectory of the given operating system path.

Although multiple Oracle installations require separate installation user IDs in BS2000 and separate Oracle home directories, you can always use the same Oracle base directory. For example:

```
/u01/app/oracbase
```

> **See Also:**
>
> "Expanding a File System for the Oracle Base Directory" for more information about the POSIX file system

2.3.6.2 About Oracle Inventory Directory

The Oracle Inventory directory (`oraInventory`) stores the inventory of all Oracle software installed on the system. It is required and shared by all Oracle software installations on a single computer. If you have an existing Oracle Inventory path, then the Oracle Database installation procedure uses that Oracle Inventory.

The Oracle Database installation procedure derives the path for the Oracle Inventory directory from the `/var/opt/oracle/oraInst.loc` file. If this file does not exist, then it derives the inventory path from `ORACLE_BASE` and creates the Oracle Inventory directory in the following path:

```
ORACLE_BASE/../oraInventory
```

For example, if `ORACLE_BASE` is set to `/u01/app/oracbase`, then the Oracle Inventory directory is created in the `/u01/app/oraInventory` path.

The Oracle Database installation procedure creates the Oracle subdirectories and sets the correct owner, group, and permissions for it. The operating system part of the

path, for example, `/u01/app` must exist and you must have read, write, and execute permissions in the `app` directory.

> **Note:**
>
> - All Oracle software installations rely on the Oracle Inventory directory. Ensure that you back it up regularly.
> - Do not delete this directory unless you have completely removed all Oracle software from the system.
> - By default, the Oracle Inventory directory is not installed under the Oracle base directory. This is because all Oracle software installations share a common Oracle Inventory, so there is only one Oracle Inventory for all software owners.

2.3.6.3 About Oracle Home Directory

The Oracle home directory is the directory where you install the software for a particular Oracle product. When you run the Oracle installation procedure, it prompts you to specify the path of this directory. You can accept the recommended path or enter a new path. The directory that you specify must be a subdirectory of the Oracle base directory. Oracle recommends that you specify a path similar to the following for the Oracle home directory:

oracle_base_path`/product/19.3.0/dbhome_1`

The Oracle installation procedure creates the directory path that you specify. It also sets the correct owner, group, and permissions. You do not have to create the subdirectories manually.

> **Note:**
>
> During the installation, you must not specify an existing directory that has predefined permissions applied to it as the Oracle home directory. If you do, then you may experience installation failure due to file and group ownership permission errors.

2.3.7 Identifying or Creating Oracle Base Directory in POSIX

Before you start the installation, you must either identify the operating system part of an existing Oracle base directory or if required, create one. This section contains information about the following topics:

- About Identifying an Existing Oracle Base Directory in POSIX
- Expanding a File System for the Oracle Base Directory
- Creating a File System for the Oracle Base Directory

> **Note:**
>
> You can create an Oracle base directory, even if other Oracle base directories exist on the system.

2.3.7.1 About Identifying an Existing Oracle Base Directory in POSIX

You can identify the existing Oracle base directories as follows:

- Identifying an existing Oracle Inventory directory.

  Search for the string `'inventory_loc'` in the file `/var/opt/oracle/oraInst.loc`:

  ```
  $ grep 'inventory_loc' /var/opt/oracle/oraInst.loc
  ```

  If `oraInst.loc` exists, then the output is similar to the following:

  ```
  inventory_loc=/u01/app/oraInventory
  ```

  The Oracle Base directory resides on the same level in the directory tree as the Oracle Inventory directory.

- Deriving an Oracle base directory from an existing Oracle home directory.

  Enter the following command to display the contents of the `/var/opt/oracle/oratab` file:

  ```
  $ more /var/opt/oracle/oratab
  ```

  If the `oratab` file exists, then it may contain lines similar to the following:

  ```
  *:/u01/app/orac1020/product/10g:N
  *:/u01/app/orac1120/product/dbhome:N
  ```

  The directory paths specified on each line identify Oracle home directories. In the preceding examples, Oracle base directories contain the user name of the Oracle software owner, `/u01/app/orac1020` and `/u01/app/orac1120` respectively. Oracle recommends to use a single base directory for all Oracle Database installations. For example:

  ```
  /u01/app/oracbase
  ```

Ensure that the Oracle base directory:

- Is not on the same file system as the operating system (`root` or `var`).
- Has sufficient free disk space.

To determine the free disk space on the file system where the Oracle base directory is located, enter the following command:

```
$ df -k oracle_base_path
```

> **Note:**
>
> The installation procedure looks for an Oracle base directory and prompts you to accept the suggested path or to enter a new path.

2.3.7.2 Expanding a File System for the Oracle Base Directory

If you want to install Oracle Database into an existing file system with existing Oracle software installations, but the POSIX file system has insufficient space, then the POSIX administrator or the BS2000 system administrator can expand the file system with the Oracle base directory using the POSIX installer. Complete the following steps to expand the file system using the POSIX installer:

1. Start the POSIX Installer using the following command:

   ```
   /START-POSIX-INSTALLATION
   ```

2. Choose **Administrate POSIX filesystems**.
3. Mark the desired file system and choose **expand**.
4. Enter the number of PAM pages by which the file system should be expanded.

2.3.7.3 Creating a File System for the Oracle Base Directory

If you want the Oracle base directory to reside in a new POSIX file system, then the POSIX administrator or the BS2000 system administrator must create a POSIX file system using the POSIX installer. Complete the following steps to create a POSIX file system using the POSIX installer:

1. Start the POSIX installer using the following command:

   ```
   /START-POSIX-INSTALLATION
   ```

2. Choose **Administrate POSIX filesystems** and then, choose **append**.
3. Enter the BS2000 file name for the container of the file system.
4. Enter the number of PAM pages of the container.

The POSIX installer allocates a new POSIX container file in the BS2000 file system and creates the POSIX file system inside this container. The POSIX installer also prompts for a mount point for the new file system. If the mount point does not exist, then it is created in the root system and the new file system is mounted. By default, the owner of the new file system is SYSROOT. The administrator must at least create the operating system part of the Oracle base directory and specify the correct owner, group, and permissions for it with the following specifications:

```
# /mkdir -p /mount_point/app/
# chown oracle_sw_owner:oracle_installation_group /mount_point/app/
# chmod 775 /mount_point/app/
```

For example:

```
# mkdir -p /u01/app/
# chown oracinst:oracle /u01/app/
# chmod 775 /u01/app/
```

2.4 About Read-Only Oracle Homes in the POSIX File System

Starting with Oracle Database 19c Release, you can install Oracle Database in a read-only Oracle Home in the POSIX file system.

Oracle Database installation in the BS2000 file system is a read-only installation by default. However, Oracle Database installation in the POSIX file system is a read/write installation by default.

Understand how read-only Oracle homes work before you choose to install read-only Oracle homes in the POSIX file system.

- **Understanding Read-Only Oracle Homes**
- About Installing a Read-Only Oracle Home in the POSIX File System
- **Determining if an Oracle Home is Read-Only**
- File Path and Directory Changes in Read-Only Oracle Homes

2.4.1 Understanding Read-Only Oracle Homes

Learn about read-only Oracle home concepts like Oracle base home, Oracle base config, and orabasetab.

- About Read-Only Oracle Homes
- **About Oracle Base Homes**
- About Oracle Base Config
- About orabasetab

2.4.1.1 About Read-Only Oracle Homes

In a read-only Oracle home, all the configuration data and log files reside outside of the read-only Oracle home.

This feature allows you to use the read-only Oracle home as a software image that can be distributed across multiple servers. In this case the installation user ID must be the same on the concerned servers.

> **Note:**
>
> An Oracle database continues to write the audit files to `$ORACLE_HOME/rdbms/audit` directory until you change the destination directory specified by the `AUDIT_FILE_DEST` initialization parameter.

Apart from the traditional ORACLE_BASE and ORACLE_HOME directories, the following directories contain files that used to be in ORACLE_HOME:

- ORACLE_BASE_HOME
- ORACLE_BASE_CONFIG

Benefits of a Read-Only Oracle Home

- Enables seamless patching and updating of Oracle databases without extended downtime.
- Simplifies patching and mass rollout as only one image needs to be updated to distribute a patch to many servers.
- Simplifies provisioning by implementing separation of installation and configuration.

2.4.1.2 About Oracle Base Homes

Both, in a read-only ORACLE_HOME and read/write ORACLE_HOME, the user-specific files, instance-specific files, and log files reside in a location known as the ORACLE_BASE_HOME.

In a read/write ORACLE_HOME, the ORACLE_BASE_HOME path is the same as the ORACLE_HOME directory. However, in a read-only ORACLE_HOME, the ORACLE_BASE_HOME directory is not co-located with ORACLE_HOME but is located at `ORACLE_BASE/homes/HOME_NAME`.

Where, HOME_NAME is the internal name for ORACLE_HOME.

For example, the networking directories `network/admin`, `network/trace`, and `network/log` are located in the ORACLE_BASE_HOME directory. In a read/write ORACLE_HOME the networking directories appear to be in ORACLE_HOME because ORACLE_BASE_HOME is co-located with ORACLE_HOME, whereas in a read-only ORACLE_HOME the networking directories are located in `ORACLE_BASE/homes/HOME_NAME`.

To print the ORACLE_BASE_HOME path, run the `orabasehome` command from the `$ORACLE_HOME/bin` directory:

```
$ ORACLE_HOME=/u01/app/oracbase/product/19.3.0/dbhome_1
$ export ORACLE_HOME
$ cd $ORACLE_HOME/bin
$ ./orabasehome
```

For example:

```
$ ./orabasehome
/u01/app/oracbase/homes/OraDB19Home1
```

Where, `/u01/app/oracbase` is ORACLE_BASE and `OraDB19Home1` is HOME_NAME.

2.4.1.3 About Oracle Base Config

Both, in a read-only ORACLE_HOME and read/write ORACLE_HOME, the configuration files reside in a location known as ORACLE_BASE_CONFIG.

In a read/write ORACLE_HOME, the ORACLE_BASE_CONFIG path is the same as the ORACLE_HOME path because it is located at $ORACLE_HOME. However, in a read-only ORACLE_HOME, the ORACLE_BASE_CONFIG path is the same as ORACLE_BASE.

`ORACLE_BASE_CONFIG/dbs` contains the configuration files for ORACLE_HOME. Each file in the `dbs` directory contains $ORACLE_SID in the path or filename so that the directory can be shared by many different ORACLE_SIDs.

To print the ORACLE_BASE_CONFIG path, run the `orabaseconfig` command from the `$ORACLE_HOME/bin` directory:

```
$ ORACLE_HOME /u01/app/oracbase/product/19.3.0/dbhome_1
$ export ORACLE_HOME
$ cd $ORACLE_HOME/bin
$ ./orabaseconfig
```

For example:

```
$ ./orabaseconfig
/u01/app/oracbase
```

Where, `/u01/app/oracbase` is ORACLE_BASE.

2.4.1.4 About orabasetab

The `orabasetab` file is used to define fundamental directories based on $ORACLE_HOME: ORACLE_BASE, ORACLE_BASE_HOME and ORACLE_BASE_CONFIG.

The `orabasetab` file resides in `ORACLE_HOME/install/orabasetab` and can be used to determine if an ORACLE_HOME is read-only or read/write. It also defines the ORACLE_BASE and the HOME_NAME of the Oracle home. HOME_NAME is the internal name for ORACLE_HOME.

The last line in the `orabasetab` file, which starts with $ORACLE_HOME, defines the directories for $ORACLE_HOME. The last line consists of four fields, each separated by a colon delimiter (:).

1. The first field matches the current $ORACLE_HOME.
2. The second field defines the ORACLE_BASE for the current ORACLE_HOME.
3. The third field defines the HOME_NAME which is used in constructing the ORACLE_BASE_HOME path in a read-only ORACLE_HOME.
4. The fourth field displays `N` in a read/write ORACLE_HOME and `Y` in a read-only ORACLE_HOME.

In a read-only ORACLE_HOME, the ORACLE_BASE_HOME path is `ORACLE_BASE/homes/HOME_NAME` and ORACLE_BASE_CONFIG is the same as ORACLE_BASE.

In a read/write ORACLE_HOME, ORACLE_HOME, ORACLE_BASE_HOME and ORACLE_BASE_CONFIG are all the same.

Viewing an orabasetab File

1. Log in as the Oracle installation owner user account.
2. Go to the `$ORACLE_HOME/install` directory.

   ```
   $ cd /u01/app/oracbase/product/19.3.0/dbhome_1/install
   ```

3. View the contents of the `orabasetab` file.

   ```
   $ cat orabasetab
   #orabasetab file is used to track Oracle Home associated with
   Oracle Base
   /u01/app/oracbase/product/19.3.0/dbhome_1:/u01/app/
   oracbase:OraDB19Home1:Y:
   ```

In this example, a `Y` in the fourth field at the end of the line indicates you have a read-only Oracle home.

2.4.2 About Installing a Read-Only Oracle Home in the POSIX File System

Starting with Oracle Database 19c, the new procedure parameter `RO-ORACLE-HOME` specifies the installation type in the POSIX file system.

Set the `RO-ORACLE-HOME` parameter to `Y` if you want a read-only ORACLE_HOME installation. Set the `RO-ORACLE-HOME` parameter to `N` if you want a read/write ORACLE_HOME installation, which is the default installation type.

If you run the installation in the DIALOG mode and you have specified an ORACLE_HOME directory for installing Oracle Database in the POSIX file system, but have not specified the type of your ORACLE_HOME directory, then you will be prompted to choose the ORACLE_HOME type in the POSIX file system.

If you run the installation in the BATCH mode and you have specified an ORACLE_HOME directory for the installation in the POSIX file system, then you must specify the type of your ORACLE_HOME by setting the procedure parameter `RO-ORACLE-HOME`. If you have not set the parameter then the default value `N` is assumed and Oracle Database will be installed in a read/write ORACLE_HOME directory.

2.4.3 **Determining if an Oracle Home is Read-Only**

Run the `orabasehome` command to determine if your Oracle home is a read/write or read-only Oracle home.

If the output of the `orabasehome` command is the same as $ORACLE_HOME, then your Oracle home is in read/write mode. If the output displays the path ORACLE_BASE/homes/HOME_NAME, then your Oracle home is in read-only mode.

1. Set the ORACLE_HOME environment variable:

```
$ ORACLE_HOME=/u01/app/oracbase/product/19.3.0/dbhome_1
$ export ORACLE_HOME
```

2. Go to the bin directory and run the `orabasehome` command:

```
$ cd $ORACLE_HOME/bin
$ ./orabasehome
/u01/app/oracbase/homes/OraDB19Home1
```

In this example, the Oracle home is in read-only mode.

2.4.4 File Path and Directory Changes in Read-Only Oracle Homes

Examples of hierarchical file mappings in a read-only Oracle home as compared to a read/write Oracle home.

This example shows an Oracle Database installation, for the user `oracinst`, with the ORACLE_HOME, ORACLE_BASE, ORACLE_BASE_HOME, and ORACLE_BASE_CONFIG locations. The database files still reside in the BS2000 file system.

This example also shows the changes in the Oracle Database software defined paths of configuration files, log files, and other directories in a read-only Oracle home when compared to a read/write Oracle home.

Table 2-1 Read-Only Oracle Home and Read/Write Oracle Home File Path Examples

Directory	Read-Only Oracle Home File Path	Read/Write Oracle Home File Path
ORACLE_HOME	`/u01/app/oracbase/product/19.3.0/dbhome_1`	`/u01/app/oracbase/product/19.3.0/dbhome_1`
ORACLE_BASE	`/u01/app/oracbase/`	`/u01/app/oracbase/`
ORACLE_BASE_HOME	ORACLE_BASE/homes/HOME_NAME (or) `/u01/app/oracbase/homes/OraDB19Home1`	ORACLE_HOME (or) `/u01/app/oracbase/product/19.3.0/dbhome_1`
ORACLE_BASE_CONFIG	ORACLE_BASE (or) `/u01/app/oracbase/`	ORACLE_HOME (or) `/u01/app/oracbase/product/19.3.0/dbhome_1`

Table 2-1 (Cont.) Read-Only Oracle Home and Read/Write Oracle Home File Path Examples

Directory	Read-Only Oracle Home File Path	Read/Write Oracle Home File Path
`network`	ORACLE_BASE_HOME/network (or) `/u01/app/oracbase/homes/OraDB19Home1/network`	ORACLE_HOME/network (or) `/u01/app/oracbase/product/19.3.0/dbhome_1/network`
`dbs`	ORACLE_BASE/dbs (or) `/u01/app/oracbase/dbs`	ORACLE_HOME/dbs (or) `/u01/app/oracbase/product/19.3.0/dbhome_1/dbs`

2.5 Installing Oracle Database

Oracle Database 19c software is available in a zipped LMS library. Complete the following steps to install Oracle Database:

1. Download the installation files from Oracle Software Delivery Cloud to a Windows or UNIX system that has FTP access to the BS2000 system where you want to install the software.

   https://edelivery.oracle.com/

2. Unzip the zip file with a zip utility, such as WinZip, to create the staging library `ora`*`xxxxx`*`.s390.lib`.

3. Transfer the staging library with the FTP binary into a user ID on the BS2000 system. This can either be the Oracle installation user ID or any other user ID. Before the transfer, if you are using BS2000 FTP, preallocate the file by using the FTP file command as shown in the example:

   ```
   file
   oraxxxxx.s390.lib,fcbtype=pam,blkctrl=no,blksize=(std,2),space=1300000
   ```

   Use the following command if you are using FTP on the platform where you had unzipped the file:

   ```
   quote file
   oraxxxxx.s390.lib,fcbtype=pam,blkctrl=no,blksize=(std,2),space=1300000
   ```

4. If you have transferred the staging library to a user ID that is not equal to the Oracle installation user ID, then you must grant access to the staging library for all users. Use the following command:

   ```
   /MODIFY-FILE-ATTRIBUTES ORAxxxxx.S390.LIB,USER-ACCESS=*ALL-USERS
   ```

5. Log in to the installation user ID.

6. If the staging library resides in the installation user ID, then call the Oracle installation procedure as follows:

```
/CALL-PROCEDURE (ORAxxxxx.S390.LIB,ORAINST.PRC)
```

If you have stored the staging library in a different BS2000 user ID, then call the Oracle installation procedure as follows:

```
/CALL-PROCEDURE (staging_userid.ORAxxxxx.S390.LIB,ORAINST.PRC),
(STAGING-UID=staging_userid)
```

For example:

```
/CALL-PROCEDURE ($FOO.ORAxxxxx.S390.LIB,ORAINST.PRC),(STAGING-
UID=$FOO)
```

Oracle Database installation procedure performs the following actions:

- Checks hardware and software requirements.
- Checks the available disk space for the BS2000 files and the POSIX files.
- Extracts the files from the staging library.
- Installs Oracle software in the installation user ID.
- Updates the SDF syntax files, if SDF-A is available.
- Installs Oracle Database software in the POSIX file system.
- Registers the software in the Oracle Inventory file.

The following is the complete syntax of the installation procedure:

```
/CALL-PROCEDURE ([staging_userid.]ORAxxxxx.S390.LIB,ORAINST.PRC)[,

([CMD={INSTALL|UNINSTALL}]

[,INSTALLATION-TYPE={DATABASE|CLIENT}]

[,STAGING-UID={''|staging_userid}]

[,ORACLE-BASE={*PROMPT|oracle_base_path}]

[,ORACLE-HOME={*PROMPT|oracle_home_path}]

[,RO-ORACLE-HOME={*PROMPT|Y|N}]

[,DEBUG={Y|N}])]
```

The following table lists the parameters, their corresponding values, and their description that are used during Oracle Database installation:

Parameter	Value	Description
CMD	INSTALL|UNINSTALL default: INSTALL	Install or uninstall Oracle Database.
INSTALLATION-TYPE	DATABASE|CLIENT default: DATABASE	Types of installation.
STAGING-UID	''|user ID of staging library default: ''	BS2000 user ID where the staging library is located.

Parameter	Value	Description
ORACLE-BASE	*PROMPT| *oracle_base_path* default: *PROMPT	The Oracle base directory is required for installing Oracle Database 19c in the POSIX file system. The default value *PROMPT, indicates the prompting for an Oracle base directory.
ORACLE-HOME	*PROMPT| *oracle_home_path* default: *PROMPT	The Oracle home directory is required for installing Oracle Database 19c in the POSIX file system. The default value *PROMPT indicates the prompting for an Oracle home directory.
RO-ORACLE-HOME	*PROMPT|Y|N default:N	Installs Oracle in the POSIX file system in READ-ONLY mode.
DEBUG	Y|N default: N	Debug the installation procedures.

Installing Oracle Database in the POSIX file system requires an ORACLE_BASE and ORACLE_HOME path. During installation you are prompted for these paths, if the installation runs in the dialog mode and if you have not specified the parameters in the procedure call. You can accept the suggested paths or enter new paths.You will also be asked if ORACLE_HOME should be installed in read-only mode or read/write mode.

The installation procedure also provides the opportunity to run the installation as a background task. In this case, you must specify the parameters ORACLE-BASE and ORACLE-HOME, otherwise the installation will be aborted if the INSTALLATION-TYPE is DATABASE. If the INSTALLATION-TYPE is CLIENT and if ORACLE-BASE or ORACLE-HOME is not specified, then the installation in the POSIX file will be skipped. If the RO-ORACLE-HOME parameter is not specified, then the default value N will be assumed. Since the installation takes about 250 CPU seconds, Oracle recommends that you set the CPU-LIMIT to at least 300 seconds. You can start the installation as follows:

```
/ENTER-PROCEDURE (ORAxxxxx.S390.LIB,ORAINST.PRC),
(ORACLE-BASE='/u01/app/oracbase',ORACLE-HOME='/u01/app/oracbase/product/
19.0.0/dbhome_1'),LOGGING=*NO,CPU-LIMIT=300
```

When you run the installation as a background task, the installation process is logged to the file INSTALL.ORAINST.LST.

2.6 Oracle Database Postinstallation Tasks

To complete the installation, the POSIX administrator or the BS2000 system administrator must run the *oracle_home_path*/root.sh script as follows:

- In the BS2000 environment, use the following command:

  ```
  /EXECUTE-POSIX-CMD CMD='oracle_home_path/root.sh'
  ```

  For example,

```
/EXECUTE-POSIX-CMD CMD='/u01/app/oracbase/product/19.3.0/dbhome_1/
root.sh'
```

- In the POSIX environment (shell), use the following command:

  ```
  # sh oracle_home_path/root.sh
  ```

  For example:

  ```
  # sh /u01/app/oracbase/product/19.3.0/dbhome_1/root.sh
  ```

If this is the first Oracle Database installation on the BS2000 server, then the `root.sh` script completes the following actions:

- Creates the path, `/var/opt/oracle`
- Creates the file, `/var/opt/oracle/oraInst.loc`
- Creates an empty file, `/var/opt/oracle/oratab`

2.7 About Installing Multiple Oracle Databases

You can install multiple Oracle databases, based on different releases of Oracle software. In this case, different releases of the software must be installed under different installation user IDs and different Oracle home directories.

2.8 About Removing Oracle Database Software

To remove Oracle Database software from your computer, log in to the Oracle installation user ID and use the installation procedure as follows:

```
/CALL-PROCEDURE INSTALL.P.ORAINST,(CMD=UNINSTALL)
```

This procedure completes the following actions:

- Removes the Oracle Database software from the POSIX file system.
- Updates the Oracle inventory.
- Removes the Oracle Database software from the BS2000 file system.

> **Note:**
>
> Only the files installed by the Oracle installation procedure, namely, `ORAINST.PRC` are removed from the system. Files created by a user or by an Oracle instance are retained in the Oracle home directory and the installation user ID.

3
About Creating a Database

Starting with Oracle Database 12*c* Release 1, you can use the Oracle Multitenant option to configure and manage a multitenant environment. The multitenant architecture enables an Oracle Database to function as a multitenant container database (CDB) that includes zero, one, or many customer-created pluggable databases (PDBs). A PDB is a portable collection of schemas, schema objects, and nonschema objects.

This chapter describes the following:

- Prerequisites for Database Creation
- About Creating a Non-CDB
- About Creating a Multitenant Container Database

3.1 Prerequisites for Database Creation

Before you can create a database, the following prerequisites must be met:

1. Oracle Database 19c must be installed under the installation user ID.
2. The BS2000 system administrator must create a DBA user ID with the required attributes.

> **See Also:**
>
> - "Oracle Database Installation and Deinstallation" for details on how to install Oracle Database
> - "About Creating Users and Groups for Oracle Databases" for details about the attributes

3.2 About Creating a Non-CDB

A non-CDB is the traditional type of an Oracle database, which was supported in Oracle Database releases before 12c.

Oracle Database 19c supports a non-CDB.

You can create a non-CDB either automatically or manually. Oracle recommends that you use the automatic creation procedure outlined in the "Creating a Database Automatically" section. Instructions on how to create a traditional database manually are given in the "Creating a Database Manually" section.

This section describes the following topics:

- Creating a Database Automatically

- Creating a Database Manually

3.2.1 Creating a Database Automatically

Complete the following steps to create a database automatically:

1. Log in to the DBA user ID.
2. To start the automatic creation procedure, `INSTALL.P.SUPER`, enter the following command:

   ```
   /CALL-PROCEDURE $ORACINST.INSTALL.P.SUPER
   ```

   When you run the `INSTALL.P.SUPER` procedure, you must specify the value of the following keyword parameters (the default values are used if you choose not to modify the values):

Parameter	Value
`BATCH`	Enter **YES** to run the procedure in the batch mode. The default is set to YES. So, by default the procedure is run in the batch mode.
`CPULIMIT`	Sets the CPU time limit for batch jobs. The default is 2000.
`PLSQL`	Enter **NO** to suppress automatic installation of the basic PL/SQL package. The default is YES.
`VIEWS`	Enter **NO** to suppress automatic installation of the basic views (catalog, import/export, and so on). The default is YES.

3. Answer the prompts for the following information. If you do not enter any value, then the default values shown on the screen are used:

Parameter	Value
`DBASID`	Enter the 1 - 4 character system ID of the database you are installing. This is a mandatory parameter.
`JOBCLASS`	Enter the jobclass to be used for Oracle Database 19c background jobs. This is mandatory. **Note:** The jobclass must be defined with the characteristics `TP-ALLOWED=YES` and `NO-CPU-LIMIT=YES`.
`UPDATE`	Enter **YES** if you have existing files for this *`SID`* and if you want to update them.
`SYSPW`	Enter the desired password for the Oracle Database user SYS.
`SYSTEMPW`	Enter the desired password for the Oracle Database user `SYSTEM`. **Note:** By default the `SYSTEM` user has the password `MANAGER`. For security reasons, Oracle recommends that you change this password immediately after installation.

Parameter	Value
JAVA	Enter **NO** if you do not need a Java enabled database (thus saving memory, CPU, and disk space resources). For more information refer to "Java in the Database" section.
DBSIZE	Enter the size of the system tablespace files in bytes, kilobytes, or megabytes. The value you enter can have one of the following forms: • 44M for 44 megabytes • 44000K for 44000 kilobytes • 10000000 for 10000000 bytes The default is 400 MB.
AUXSIZE	Enter the size of the sysaux tablespace file in bytes, kilobytes, or megabytes. The value you enter can have one of the following forms: • 44M for 44megabytes • 44000K for 44000 kilobytes • 10000000 for 10000000 bytes The default is 300 MB.
LOGSIZE	Enter the size of the log files in bytes, kilobytes, or megabytes. The value you enter can have one of the following forms: • 1M for 1 megabytes • 1000K for 1000 kilobytes • 100000 for 100000 bytes The default is 20000 KB.
UNDOSIZE	Enter the size of the undo tablespace file in bytes, kilobytes, or megabytes. The value you enter can have one of the following forms: • 44M for 44 megabytes • 44000K for 44000 kilobytes • 10000000 for 10000000 bytes The default is 100 MB.
LOCAL	Enter **NO** if you do not require a locally managed system tablespace. The default is YES. If you choose a locally managed system tablespace, then Oracle automatically creates a default temporary tablespace.
DEFTS	Enter **NO** if you do not want to create a default permanent tablespace. The default is YES.
TEMPTS	This prompt only appears if you do not want a locally managed system tablespace. Enter **NO** if you do not want a default temporary tablespace. The default is YES.
CHARSET	Enter the character set with which you want the database to be created (the default is WE8BS2000). For more information refer to "Globalization Support".
NCHARSET	Enter the national character set used to store data in columns specifically defined as NCHAR, NCLOB, or NVARCHAR2. Valid values are AL16UTF16 and UTF8. The default is AL16UTF16.

Unless specified otherwise, $ORACINST.INSTALL.P.SUPER generates and enters a batch job which:

- Calls INSTALL.P.DBA.
- Creates the system and sysaux tablespace.
- Creates the default permanent tablespace, temporary tablespace, and undo tablespace.

- Creates the log files.
- Initializes the database.
- Runs `CATALOG.SQL`.
- Runs `CATPROC.SQL` .
- Installs the tables for online help messages.
- Installs the `DEMO` tables.
- Changes the system passwords if necessary.
- Calls the verification procedure.

When `$ORACINST.INSTALL.P.SUPER` has completed, you should have an initialized, ready-to-use database, and a running Oracle Database instance. The results of the job are listed in the file, `L.`*`sid`*`.INSSUP.SYSOUT`, where *`sid`* is the system ID of the database that you just installed.

3.2.2 Creating a Database Manually

Oracle recommends that you use the automatic creation procedure outlined in "Creating a Database Automatically". The manual creation procedure performs the same steps as the automatic creation procedure. However, when you create a database manually, you can perform the steps at your own pace, and also choose which of the optional steps to perform, omit, or perform later.

This section describes the following topics:

- Creating Parameter Files for a Non-CDB
- Creating the Database
- About Installing Data Dictionary Views
- About Installing Online Help Messages
- About Installing Demo Tables
- About Installing Sample Schemas
- About Verifying Database Creation
- About Installing Oracle Text
- About Installing Java

3.2.2.1 Creating Parameter Files for a Non-CDB

Create the parameter files for the new database as follows:

1. Log in to the DBA user ID.
2. Call the BS2000 command procedure `INSTALL.P.DBA`. This procedure generates parameter files for the database in the DBA user ID. When the procedure begins you are prompted to supply a 1 to 4 character system ID for the database you are creating.

   To install the DBA files, enter the following command:

   ```
   /CALL-PROCEDURE $ORACINST.INSTALL.P.DBA
   ```

   This procedure prompts you for the following information:

Parameter	Value
DBASID	Enter the 1 - 4 character system ID of the database you are creating.
JOBCLASS	Enter the BS2000 jobclass to be used for background and server tasks. The jobclass must be defined with the characteristics TP-ALLOWED=YES and NO-CPU-LIMIT=YES.

You can also modify the following keyword parameters when invoking this procedure:

Parameter	Value
LOG	Enter **WRITE-TEXT** (the BS2000 command name) if you want to have install actions listed.
UPDATE	Enter **YES/NO** to indicate whether existing files are to be updated. The default is NO.

The $ORACINST.INSTALL.P.DBA procedure generates the following files into the DBA user ID:

- *sid*.P.ORAENV: Oracle Database environment definition file
- *sid*.DBS.INIT.ORA: Oracle Database initialization file

where *sid* is the system ID for the database being created.

These files are generated by using the two corresponding files DEMO.P.ORAENV and DEMO.DBS.INIT.ORA in the installation user ID as templates.

3.2.2.2 Creating the Database

After generating the DBA files, you must create the database. This section describes the procedure for creating the database:

- About Modifying the Initialization File for a Non-CDB
- About Modifying the ORAENV File for a Non-CDB
- About Using SQL*Plus to Create the Database

3.2.2.2.1 About Modifying the Initialization File for a Non-CDB

Determine if you want to make any changes to parameters in the distributed initialization file, *sid*.DBS.INIT.ORA (where *sid* is the system ID for the database). The SGA parameters may need to be adjusted to reflect memory limitations and the maximum number of users who can access the Oracle Database instance concurrently. Use a BS2000 editor to make the modifications.

> **See Also:**
>
> *Oracle Database Reference* for more information about the initialization parameters

3.2.2.2.2 About Modifying the ORAENV File for a Non-CDB

Modify the environment definition file, *sid*.P.ORAENV, according to your specific requirements. Remember that a number of variables are evaluated during startup only. If you modify such a variable in the ORAENV file later on, then you may have to wait for the next startup for the changes to take effect.

Do not explicitly set the environment variable NLS_LANG when you run Oracle supplied SQL scripts.

3.2.2.2.3 About Using SQL*Plus to Create the Database

Remember that you must call the applicable *sid*.P.ORAENV procedure before calling SQL*Plus. To execute SQL*Plus, enter the following commands:

```
/START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
* /NOLOG
SQL> CONNECT / AS SYSDBA
SQL> STARTUP NOMOUNT [PFILE=filename]
```

- /NOLOG omits being prompted for user name and password.
- CONNECT gives you a connection to an idle instance.
- The last statement starts the Oracle Database instance.

If you want to use your own copy of the initialization file (*sid*.DBS.INIT.ORA), then use the PFILE=*filename* option, as illustrated in the previous command.

The following statement creates database and log files:

```
SQL> CREATE DATABASE...;
```

> **Note:**
>
> If you get an error before the first SQL> prompt, then it may be caused by either a missing ORAENV file (or ORASID not set in the ORAENV file), or sometimes by an address space conflict. For example, the address range you assigned to the kernel memory pool (KNL_BASE) could be occupied by a subsystem.

3.2.2.3 About Installing Data Dictionary Views

Run the scripts necessary to build data dictionary views, synonyms, and PL/SQL packages. Run the following commands:

```
SQL> SPOOL filename
SQL> SET TERMOUT OFF
SQL> $ORACLE_HOME/rdbms/admin/catalog.sql
SQL> $ORACLE_HOME/rdbms/admin/catproc.sql
```

3.2.2.4 About Installing Online Help Messages

To install the Online Help facility, enter the following command:

```
/CALL-PROCEDURE $ORACINST.INSTALL.P.HELP,(sid[,SYSTEMPW=systempw])
```

3.2.2.5 About Installing Demo Tables

To install demo tables, enter the following:

```
/CALL-PROCEDURE $ORACINST.INSTALL.P.DEMO,(sid [,SYSTEMPW=systempw])
```

3.2.2.6 About Installing Sample Schemas

To install sample schemas, enter the following:

```
/CALL-PROCEDURE $ORACINST.INSTALL.P.SAMPLES,(sid [,SYSTEMPW=systempw]
[,SYSPW=syspw])
```

The procedure `INSTALL.P.SAMPLES` installs the human resources (HR), order entry (OE), info exchange (IX), and sales history (SH) sample schemas with the default passwords. Product media (PM) is not supported.

3.2.2.7 About Verifying Database Creation

To verify if the demonstration database was correctly created, enter the following:

```
/CALL-PROCEDURE $ORACINST.INSTALL.P.VERIFY,(sid [,SYSTEMPW=systempw])
```

If the demonstration database was correctly created, then you see messages like the following displayed on the screen:

```
*SCOTT'S TABLE EMP IS INSTALLED
```

3.2.2.8 About Installing Oracle Text

Installing Oracle Text is summarized in the "Oracle Text" chapter of this book.

3.2.2.9 About Installing Java

Installing Java is summarized in the "Java in the Database" chapter of this book.

3.3 About Creating a Multitenant Container Database

A multitenant container database (CDB) is a single physical database that contains zero, one, or many user-created pluggable databases. A pluggable database (PDB) is a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a non-CDB. A non-CDB is a traditional Oracle database that cannot contain PDBs.

Starting with Oracle Database 12*c* Release 1, you can create a multitenant container database (CDB).

The topics in this section describe how to create a CDB manually on Fujitsu BS2000. As creating a CDB involves Perl scripts, you must perform some steps in the POSIX environment. Perl is supported only in the POSIX environment of BS2000.

The following topics are discussed:

- Creating Parameter Files for a CDB

- About Creating a CDB

> **See Also:**
>
> *Oracle Database Administrator's Guide* for more information about creating a CDB

3.3.1 Creating Parameter Files for a CDB

Create the parameter files for the new database as follows:

1. Log in to the DBA user ID.
2. Call the BS2000 command procedure `INSTALL.P.DBA`. This procedure generates parameter files for the database in the DBA user ID. When the procedure begins you are prompted to supply a 1 to 4 character system ID for the database you are creating.

   To install the DBA files, enter the following command:

   ```
   /CALL-PROCEDURE $ORACINST.INSTALL.P.DBA
   ```

   This procedure prompts you for the following information:

Parameter	Value
`DBASID`	Enter the 1 - 4 character system ID of the database you are creating.
`JOBCLASS`	Enter the BS2000 jobclass to be used for background and server tasks. The jobclass must be defined with the characteristics `TP-ALLOWED=YES` and `NO-CPU-LIMIT=YES`.

   You can also modify the following keyword parameters when invoking this procedure:

Parameter	Value
`LOG`	Enter **WRITE-TEXT** (the BS2000 command name) if you want to have install actions listed.
`UPDATE`	Enter **YES/NO** to indicate whether existing files are to be updated. The default is NO.

   The `$ORACINST.INSTALL.P.DBA` procedure generates the following files into the DBA user ID:

   - *sid*`.P.ORAENV`: Oracle Database environment definition file
   - *sid*`.DBS.INIT.ORA`: Oracle Database initialization file

   where *sid* is the system ID for the database being created.

   These files are generated by using the two corresponding files `DEMO.P.ORAENV` and `DEMO.DBS.INIT.ORA` in the installation user ID as templates.

3.3.2 About Creating a CDB

The procedure to create a multitenant container database (CDB) is discussed in the following topics:

- About Modifying the Initialization File for a CDB
- About Modifying the ORAENV File for a CDB
- Using SQL*Plus to Create a CDB

3.3.2.1 About Modifying the Initialization File for a CDB

Determine if you want to make any changes to the parameters in the initialization file, *sid*.DBS.INIT.ORA (where *sid* is the system ID for the database). The SGA parameters may need to be adjusted to reflect memory limitations and the maximum number of users who can access the Oracle Database instance concurrently. Use a BS2000 editor to make the modifications.

To create a CDB, set the value for the ENABLE_PLUGGABLE_DATABASE initialization parameter in the *sid*.DBS.INIT.ORA file as follows:

```
ENABLE_PLUGGABLE_DATABASE=TRUE
```

> **See Also:**
>
> *Oracle Database Reference* for an explanation of initialization parameters

3.3.2.2 About Modifying the ORAENV File for a CDB

Modify the environment definition file, *sid*.P.ORAENV, according to your specific requirements. Remember that a number of variables are evaluated during startup only. If you modify such a variable in the ORAENV file later on, then you may have to wait for the next startup for the changes to take effect.

Do not explicitly set the environment variable NLS_LANG when you run Oracle supplied SQL scripts.

3.3.2.3 Using SQL*Plus to Create a CDB

Complete the following steps to create a CDB using the SQL*Plus utility:

1. In the DBA user ID, change to the POSIX shell by executing the BS2000 SDF command:

   ```
   /START-POSIX-SHELL
   ```

2. Start SQL*Plus in the POSIX shell and start up the instance with the NOMOUNT option:

   ```
   $ . oracle_home_path/.profile.oracle
   $ ORACLE_SID=sid
   $ export ORACLE_SID
   $ sqlplus /nolog
   ```

```
SQL> connect / as sysdba
SQL> startup nomount;
```

3. Execute a SQL statement similar to the following to create `CDB$ROOT` and `PDB$SEED` of the CDB. In the following example, a CDB with the name `CDB1` is created:

```
SQL> CREATE DATABASE CDB1
USER SYS IDENTIFIED BY sys_password
USER SYSTEM IDENTIFIED system_password
LOGFILE 'CDB1.DBS.LOG1.DBF' SIZE 20M, 'CDB1.DBS.LOG2.DBF' SIZE 20M
DATAFILE 'CDB1.DBS.DATABASE1.DBF' SIZE 400M
SYSAUX DATAFILE 'CDB1.DBS.SYSAUX.DBF' SIZE 200M
DEFAULT TABLESPACE USERS
DATAFILE 'CDB1.DBS.USERS1.DBF' SIZE 100M AUTOEXTEND ON
DEFAULT TEMPORARY TABLESPACE TEMP
TEMPFILE 'CDB1.DBS.TEMP1.DBF' SIZE 100M AUTOEXTEND ON
UNDO TABLESPACE UNDOTBS1
DATAFILE 'CDB1.DBS.UNDO1.DBF' SIZE 100M AUTOEXTEND ON
CHARACTER SET WE8BS2000
NATIONAL CHARACTER SET AL16UTF16
EXTENT MANAGEMENT LOCAL
ENABLE PLUGGABLE DATABASE
SEED FILE_NAME_CONVERT = ('DBS','SEED');
```

4. The next statement installs all of the components required by a CDB. For example, it runs the SQL scripts `catalog.sql` and `catproc.sql` in `CDB$ROOT` and `PDB$SEED` of the CDB:

```
SQL> @$ORACLE_HOME/rdbms/admin/catcdb.sql
```

5. Exit SQL*Plus and check the spool files for error messages.

4
About Upgrading a Database

This chapter describes how to upgrade your existing database to Oracle Database 19c on Fujitsu BS2000. It gives you an overview about the steps required for the upgrade of a database and informs about actions, which are specific to the Fujitsu BS2000 platform.

Before you perform an upgrade, you must be familiar about upgrade preparation, space and backup requirements, release differences, handling `TIMESTAMP WITH TIME ZONE` data type, and other database upgrade concepts.

It is assumed that you have correctly installed Oracle Database 19c as explained in "Oracle Database Installation and Deinstallation."This chapter describes upgrading a database from Oracle Database 12*c* Release 2 (12.2) to Oracle Database 19c. The upgrade is done both in the normal BS2000 environment and in the POSIX shell. Oracle recommends that you be familiar with both these environments.

This chapter includes the following topics:

- Performing Preupgrade Procedures
- Performing Upgrade Procedures
- Performing Postupgrade Procedures

Related Topics

- *Oracle Database Upgrade Guide*

4.1 Performing Preupgrade Procedures

You must analyze your database before upgrading it to the new release. To analyze, run the `preupgrade.jar` Pre-Upgrade Information Tool from the environment of the database that you want to upgrade. Running the Pre-Upgrade Information Tool provides information about any issues that need to be fixed.

You must run the `preupgrade.jar` tool in the environment of the source database, for example, in the Oracle Database 12c Release 2 (12.2) environment. The database must be up and running.

The Pre-Upgrade Information Tool generates log files and fixup scripts that you can run, to resolve issues that are flagged in the source database. These log files and fixup scripts are generated in the POSIX file system. If `ORACLE_BASE` is defined, then the generated scripts and log files are created in the `$ORACLE_BASE/cfgtoollogs` directory. If `ORACLE_BASE` is not defined, then the generated scripts and log files are created in the `$ORACLE_HOME/cfgtoollogs` directory.

Perform the following steps in the environment of the source database:

1. Log in to the DBA user ID of the source database.
2. To avoid being prompted for many overflow acknowledgements on your screen, enter:

   ```
   /MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=*NO-CONTROL
   ```

3. Change to the POSIX shell:

   ```
   /START-POSIX-SHELL
   ```

4. Before you run the Pre-upgrade Information Tool, `preupgrade.jar` in the POSIX shell, set the Oracle environment variables `ORACLE_HOME` and `ORACLE_BASE` by executing the `.profile.oracle` file:

   ```
   $ . oracle_home_path_12c_R2/.profile.oracle
   ```

   You must also set the `ORACLE_SID` environment variable to the *`sid`* of your Oracle Database instance. Use the following commands:

   ```
   $ ORACLE_SID=sid
   $ export ORACLE_SID
   ```

   > **Note:**
   >
   > You must remember that the variable `BGJPAR` might not be defined. Oracle recommends to set the variable `BGJPAR` according to the value you find in the corresponding `ORAENV` file , *`sid`*`.P.ORAENV`.

5. Run the `preupgrade.jar` Pre-Upgrade Information Tool:

   ```
   $ $ORACLE_HOME/jdk/bin/java -jar oracle_home_path_19c/rdbms/admin/
   preupgrade.jar
   ```

6. View and read through the resulting generated fixup scripts and log file, which are located in:

   - `$ORACLE_BASE/cfgtoollogs/`*`db_unique_name`*`/preupgrade` directory if `ORACLE_BASE` is defined.
   - `$ORACLE_HOME/cfgtoollogs/`*`db_unique_name`*`/preupgrade` directory if `ORACLE_BASE` is not defined.

7. After you have reviewed the scripts, Oracle recommends that you execute `preupgrade_fixups.sql` on the source database. The `preupgrade_fixups.sql` script attempts to resolve issues reported by the preupgrade process. You must manually resolve the issues that cannot be resolved automatically by a fixup script.

8. Shutdown the database and exit SQL*Plus. Use the following commands:

   ```
   $ sqlplus /nolog
   SQL> connect / as sysdba
   SQL> shutdown immediate;
   SQL> exit
   ```

9. Exit the POSIX shell:

   ```
   $ exit
   ```

> **Note:**
>
> Oracle recommends that you back up your Oracle database after running the Pre-Upgrade Information Tool and shutting down the database.
>
> You can find regular updates of the Pre-Upgrade Information tool on the My Oracle Support website. Refer to the My Oracle Support Note 884522.1 if you want to download and install a new release of the Pre-Upgrade information tool. The steps described in this note are valid for Fujitsu BS2000, too. The only difference is the procedure for unzipping the zip file.
>
> Transfer the zip file (with binary mode) to the POSIX file system of your BS2000 machine. Then unzip the zip file into the `$ORACLE_HOME/rdbms/admin` directory of Oracle Database 19c as follows:
>
> ```
> $ cd $ORACLE_HOME/rdbms/admin
> $ unzip -aa preupgrade_19_cbuild_nn_lf.zip -x preupgrade.jar
> $ unzip preupgrade_19_cbuild_nn_lf.zip preupgrade.jar
> ```

4.2 Performing Upgrade Procedures

Use the upgrade utility, `dbupgrade` to upgrade Oracle Database. As shell scripts and Perl scripts are supported only in the POSIX environment, you must change to the POSIX shell for performing some of the procedure.

The shell script `dbupgrade` is a wrapper for the parallel upgrade utility `catctl.pl`, which is based on Perl.

All the steps in this phase must be performed in the new Oracle Database 19c environment. For some steps you have to change to the POSIX shell.

Perform the following upgrade steps:

1. Log in to the DBA user ID of the database to be upgraded.
2. To avoid being prompted for many overflow acknowledgements on your screen, enter:

   ```
   /MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=*NO-CONTROL
   ```

3. Enter the following command to create an Oracle Database 19c `INIT.ORA` file and an Oracle Database 19c `ORAENV` file. The original files will be saved by the `$ORACINST.INSTALL.P.DBA` procedure with the suffix `.OLD`. Use the following commands:

   ```
   /CALL-PROCEDURE $ORACINST.INSTALL.P.DBA,(sid,jobclass,UPDATE=YES)
   ```

4. Modify the newly created files according to your requirements. For example, change the values of `PROCESSES` or `DB_CACHE_SIZE` in the `INIT.ORA` file.
5. Set the parameters in the `INIT.ORA` file as recommended by the Pre-Upgrade Information Tool. Ensure that the `COMPATIBLE` initialization parameter is explicitly set to 12.2.0 or higher.

6. Change to the POSIX shell. Use the following command:

   ```
   /START-POSIX-SHELL
   ```

7. Before you run SQL*Plus in the POSIX shell you must set some environment variables. For example, `ORACLE_HOME` and `ORACLE_BASE`. To set these variables, run the `.profile.oracle` file that is located in the `ORACLE_HOME` directory of the Oracle Database 19c software installation:

   ```
   $ . oracle_home_path_19c/.profile.oracle
   ```

8. Set the `ORACLE_SID` environment variable to the *sid* of your Oracle Database instance:

   ```
   $ ORACLE_SID=sid
   $ export ORACLE_SID
   ```

   > **Note:**
   >
   > Ensure that you define the `BGJPAR` variable. Oracle recommends to set the `BGJPAR` variable to the corresponding value in the `ORAENV` file, *sid*`.P.ORAENV`.

9. Start up the instance in upgrade mode as follows:

   ```
   $ sqlplus /nolog
   SQL> connect / as sysdba
   SQL> startup upgrade
   SQL> exit
   ```

10. Run the shell script for upgrade as follows:

    ```
    $ $ORACLE_HOME/bin/dbupgrade -l $HOME/logs
    ```

    The `-l` option allows you to specify the directory where you want to write the upgrade log files.

    > **See Also:**
    >
    > *Oracle Database Upgrade Guide* for other options of the upgrade utility, `dbupgrade`.

    The upgraded database is shut down after running `dbupgrade`.

11. Restart the instance to reinitialize the system parameters for normal operation:

    ```
    $ sqlplus /nolog
    SQL> connect / as sysdba
    SQL> startup
    ```

12. Run the `catcon.pl` script to start `utlrp.sql`, and to recompile any remaining stored PL/SQL and Java code. For example:

```
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlrp -l $HOME/logs -d
'''.''' utlrp.sql
```

When you run the command using `-b utlrp`, the log file `utlrp0.log` is generated as the script is run. The log file provides the recompilation results.

13. Run the `postupgrade_fixups.sql` script. For example:

```
SQL> @postupgrade_fixups.sql
```

14. Run `utlusts.sql`, the Post-Upgrade Status Tool to display a summary of the upgrade results:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlusts.sql TEXT
```

If the `utlusts.sql` script returns errors, or shows components that do not contain the most recent release, or do not have the `VALID` status, then refer to *Oracle Database Upgrade Guide* for troubleshooting.

15. To query invalid objects, execute SQL queries similar to:

```
SQL> SELECT count(*) FROM dba_objects WHERE status='INVALID';
SQL> SELECT distinct object_name FROM dba_objects WHERE
status='INVALID';
```

Your database is now upgraded to Oracle Database 19c.

4.3 Performing Postupgrade Procedures

You must complete the following tasks after upgrading to Oracle Database 19c:

- Gathering Statistics
- Migrating to Unified Auditing

Gathering Statistics

Oracle recommends to gather dictionary statistics after upgrading the database. Execute the following SQL statement:

```
SQL> execute dbms_stats.gather_dictionary_stats;
```

After an upgrade Oracle strongly recommends that you regather fixed object statistics after you have run representative workloads. Execute the following SQL statement:

```
SQL> execute dbms_stats.gather_fixed_objects_stats;
```

Migrating to Unified Auditing

Oracle Database 19c supports Unified Auditing. After upgrading to Oracle Database 19c, the Unified Auditing option is not enabled.

See Also:

- "About Unified Auditing" for more information about enabling Unified Auditing
- *Oracle Database Upgrade Guide* for more information about migrating to Unified Auditing

5
About Upgrading Applications

Learn how to rebuild Application Programs in the following topics.

- **Precompile and Compile Application Programs**
- Link Application Programs
- Update ORAENV Files

Related Topics

- *Oracle Database Upgrade Guide*

5.1 Precompile and Compile Application Programs

You must change the Application code when APIs are deprecated or changed.

When you upgrade to the new release of Oracle Database software, ensure that you:

- Make the necessary code changes for APIs that are either deprecated or changed.
- Precompile and compile embedded SQL applications.
- Compile OCI applications with the new software.
- Relink the application using the libraries of the new release of Oracle Database.

By recompiling, you perform a syntax check of your application code. Some problems in the application code that were not detected by previous releases of the Oracle software can emerge when you precompile or compile with the new Oracle Database software. Precompiling and compiling with the latest release of Oracle Database software helps you to detect and correct problems in the application code that were previously unnoticed.

5.2 Link Application Programs

Statically-linked code can be incompatible with the upgraded Oracle Software.

It is very important that you relink statically-linked applications using the libraries of the new release of Oracle Database Software.

5.3 Update ORAENV Files

Check your `ORAENV` files, and if necessary, update the value of the `ORAUID`, `ORACLE_BASE`, and `ORACLE_HOME` environment variables.

The `ORAUID` environment variable must refer to the correct Oracle Database installation user ID and the `ORACLE_HOME` environment variable must refer to the corresponding Oracle home directory in the POSIX file system. Also check other environment variables and update them as required.

Use the `INSTALL.P.USER` procedure to create a template of an `ORAENV` file for the new release of Oracle Database.

Part III
Database Administration

This part discusses the following topics related to Oracle Database administration:

- Administering Oracle Database
- Oracle Database Utilities
- Backing Up and Recovering a Database
- About Unified Auditing
- Java in the Database
- Oracle Text
- XML
- Oracle Net Services

6
Administering Oracle Database

This chapter describes how to administer Oracle Database 19c for BS2000.

Common administration tasks are described in the following sections:

- Using the SQL*Plus Utility
- Startup and Parameter Files
- Preparing a Remote Startup of a Database Instance Using SQL*Plus
- Automatic Diagnostic Repository

6.1 Using the SQL*Plus Utility

SQL*Plus is the primary command-line interface to administer your Oracle database. You can use SQL*Plus to start up and shut down the database, set database initialization parameters, create and manage users, create and alter database objects, and so on.

> **See Also:**
>
> "SQL*Plus" for more information about how to use SQL*Plus on Fujitsu BS2000

6.2 Startup and Parameter Files

Oracle uses the following parameter files when starting the database:

1. The environment definition file `ORAENV`, which contains BS2000-specific information. In the `ORAENV` file, identify the database that has to be started or shut down. You can use this file to set the required environment variables.
2. The initialization file `INIT.ORA` or the server parameter file `SPFILE`, which exists in all Oracle Database implementations and contains database-specific parameters.

This section describes the following:

- The Environment Definition File ORAENV
- The Initialization File INIT.ORA
- The Server Parameter File SPFILE
- About Using the Initialization File

6.2.1 The Environment Definition File ORAENV

The `ORAENV` file is identified by *`sid`*`.P.ORAENV`, where *`sid`* is the system identifier. The same `ORAENV` file must be used by SQL*Plus in BS2000 and by all background jobs.

If you use SQL*Plus in the POSIX shell, then you must specify the requested BS2000 parameters that are set in the ORAENV file. You can set the variables in the POSIX environment or use the facility to access the BS2000 ORAENV file as described in "Setting Variable in the POSIX Program Environment."

Ensure that the value for *sid* in the POSIX file name oraenv*sid* matches the value of the environment variable ORACLE_SID. For example, if you created a POSIX file oraenv*sid* with *sid* in uppercase, then you must set the environment variable ORACLE_SID to exactly the same value.

```
$ echo '$ORADATA.ORCL.P.ORAENV' > $ORACLE_HOME/dbs/oraenvORCL
$ ORACLE_SID=ORCL
$ export ORACLE_SID
$ sqlplus /nolog
$ SQL> connect / as sysdba
```

See Also:

- "Oracle Environment Variables" for information about required and optional ORAENV variables
- "Starting Oracle Utilities in the POSIX Program Environment" for more information about how to set POSIX environment variables

6.2.2 The Initialization File INIT.ORA

To start up a database, you require the INIT.ORA parameter file. This file contains a list of specifications for the Oracle database. The platform independent parameters set up the instance and the database.

Related Topics

- *Oracle Database Administrator's Guide*
- *Oracle Database Reference*

6.2.3 The Server Parameter File SPFILE

You can choose to maintain initialization parameters in a binary server parameter file. A server parameter file is initially built from a traditional text initialization parameter file using the CREATE SPFILE command. If you enter the following command:

```
CREATE SPFILE FROM PFILE;
```

where neither SPFILE name nor PFILE name is specified, then Oracle looks for a text initialization file *sid*.DBS.INIT.ORA and creates a server parameter file *sid*.DBS.SPFILE.ORA.

6.2.4 About Using the Initialization File

A default initialization file, called $ORACINST.DEMO.DBS.INIT.ORA, is distributed with Oracle Database. During database creation, this file is copied to the DBA user ID and renamed to *sid*.DBS.INIT.ORA, where *sid* is the 1 to 4 character system ID you specified at the beginning of the database creation procedure.

Oracle determines the value of *sid* by retrieving the ORASID environment variable defined in the ORAENV file for the database. When you issue the STARTUP command without specifying the PFILE, Oracle locates the initialization parameter file by examining file names in the following order:

1. *sid*.DBS.SPFILE.ORA
2. DBS.SPFILE.ORA
3. *sid*.DBS.INIT.ORA

If you want to use a different initialization file, then use the argument PFILE. For example, to bring up a previously created database using an initialization file called TEST.INIT.ORA, enter the following:

```
/START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
* /NOLOG
```

At the SQL*Plus prompt, enter:

```
SQL> CONNECT / AS SYSDBA
SQL> STARTUP PFILE=TEST.INIT.ORA
```

6.3 Preparing a Remote Startup of a Database Instance Using SQL*Plus

This section describes the preparations that are required to start up a database instance remotely using SQL*Plus:

1. Usually, the listener parameter file, LISTENER.ORA does not contain a static service registration section (SID_LIST) for a database service. In case of a remote startup, you must define this section for the desired database. For example:

   ```
   SID_LIST_LISTENER = (SID_LIST =
                        (SID_DESC =
                         (SID_NAME = ORCL)))
   ```

   The listener must be running on the computer where you want to start the instance. The listener must statically register the instance. If the listener does not run under the same user ID as the instance that you want to start, then you must do one of the following:

   - Use the Oracle environment variables *sid*_USER, *user*_ACCOUNT, and *user*_PASSWORD to specify the required LOGON authorization parameters in the ORAENV file of the listener.
   - Use SECOS, Fujitsu's Security Control System for BS2000.

   > **See Also:**
   >
   > "Configuring the Network" for more information

2. Create a password file with the Oracle utility, ORAPWD under the user ID of the instance that you want to administer. To run the ORAPWD utility on BS2000, use the following command:

```
/START-EXECUTABLE (*LINK(ORALOAD),ORAPWD)
*file=password_file password=my_password entries=10
```

> **See Also:**
>
> *Oracle Database Administrator's Guide* for more information about how to use the `ORAPWD` utility

3. The name of the password file is derived from the `SSSIDPWF` environment variable. Ensure that you add this environment variable to the `ORAENV` file of the instance that you want to start:

   ```
   SSSIDPWF = password_file
   ```

4. The parameter `REMOTE_LOGIN_PASSWORDFILE` must be set to `EXCLUSIVE` in the initialization file of the instance:

   ```
   REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE
   ```

5. Execute SQL*Plus on the remote computer and connect as user `sys` to a server task of the instance that you want to start. The following example shows the commands for SQL*Plus on a UNIX client. The net service name *orcl_on_bs2000* is used to address the remote instance on the BS2000 computer:

   ```
   sqlplus /nolog
   SQL> connect sys@orcl_on_bs2000 as sysdba
   Enter password:
   password
   Connected
   SQL> startup
   ...
   ```

6.4 Automatic Diagnostic Repository

Automatic Diagnostic Repository (ADR) is a file-based hierarchical data store for depositing diagnostic information produced by diagnostic framework clients. The repository contains data describing incidents, traces, dumps, alert logs, health check records, SQL Trace information, and other information essential for problem diagnosis.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for more information about Automatic Diagnostic Repository

This section describes the following:

- Automatic Diagnostic Repository Directories and Files
- ADR Command Interpreter Utility

6.4.1 Automatic Diagnostic Repository Directories and Files

Automatic Diagnostic Repository (ADR) is a directory structure that is stored outside of the database. It is therefore available for problem diagnosis when the database is down.

The directories and files of the Automatic Diagnostic Repository are stored in the POSIX file system.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about the directory structure

The ADR root directory is known as ADR base. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter. For example:

```
DIAGNOSTIC_DEST=/u01/app/oracbase/oradata/adr
```

If this parameter is omitted or left null, then the database sets the `DIAGNOSTIC_DEST` parameter upon startup as follows:

- If the environment variable `ORACLE_BASE` is set, then the `DIAGNOSTIC_DEST` parameter is set to the directory designated by `ORACLE_BASE`.
- If the environment variable `ORACLE_BASE` is not set, then the `DIAGNOSTIC_DEST` parameter is set to `ORACLE_HOME/log`.

Within ADR base, there can be multiple ADR homes, where each ADR home is the `root` directory for all diagnostic data, such as, traces, dumps, alert log, and so on, for a particular instance of a particular Oracle product or component.

Oracle Net Services also stores diagnostic data in the ADR. The location for diagnostic information for Oracle Net Services is set by the `ADR_BASE` and `ADR_BASE_`*`listener_name`* parameters. These are set in the `sqlnet.ora` and `listener.ora` Oracle Net Services parameter files.

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

You can read the text files of the Automatic Diagnostic Repository with text editors such as `vi` or `edtu`, or with POSIX shell commands such as `cat` or `more`.

You can investigate ADR with the ADRCI utility.

6.4.2 ADR Command Interpreter Utility

ADR Command Interpreter (ADRCI) utility enables you to investigate problems, view health check reports, and package first-failure diagnostic data within a command-line environment. You can then upload the package to Oracle Support. ADRCI also enables you to view the following:

- Names of the trace files in the ADR.
- Alert log with XML tags stripped, with and without content filtering.

You must execute the ADRCI command-line utility in the POSIX shell. Before you start the ADRCI utility, you must set the `ORACLE_HOME` environment variable. The system variable `PATH` must be extended by the path name of the Oracle bin directory, `$ORACLE_HOME/bin`. You can set the environment variables by executing the `.profile.oracle` profile in the appropriate Oracle home directory. Additional environment variables, such as `ORACLE_SID`, are not required.

> **See Also:**
>
> - *Oracle Database Utilities* for more information about ADRCI
> - "Starting Oracle Utilities in the POSIX Program Environment" for more information about Oracle Database utilities

After setting these environment variables, start the ADRCI utility in the POSIX shell by entering `adrci` after the POSIX shell command prompt.

> **Note:**
>
> You cannot start the ADRCI utility in the BS2000 program environment. ADRCI utility must be started in the POSIX shell.

Display the current ADR base with the ADRCI `show base` command. The current ADR home can be displayed with the `show homes` command.

Set ADR base with the `set base` command. You can set ADR home with the `set home` command.

The `show alert` command shows the contents of the alert log in a text editor.

If you use a blockmode terminal, then the default editor for ADRCI on Fujitsu BS2000 is `edtu`. If you use an xterm terminal, after having logged in to POSIX through `rlogin` or `ssh`, then the default editor for ADRCI is `vi`.

You can either select the preferred text editor by setting the environment variable `EDITOR` before starting the ADRCI utility or specify your preferred text editor within ADRCI with the ADRCI `set editor` command.

> **Note:**
>
> The editor `vi` does not work on blockmode terminals and the editor `edtu` does not work on xterm terminals.

With ADRCI, you can invoke Incident Packaging Service (IPS) to create packages for incidents with the following commands:

```
ips create package
ips generate package
```

Upload the resulting zip file to Oracle Support.

7

Oracle Database Utilities

The various Oracle Database utilities and how to use them with Fujitsu BS2000 are discussed in the following topics:

- Basics of Oracle Database Utilities
- SQL*Plus
- The SQL*Loader
- The Export Utility
- The Import Utility
- The Data Pump Export Utility
- The Data Pump Import Utility
- Recovery Manager on BS2000
- Checking the Integrity of the Physical Data Structure
- Workload Replay Client
- The Oracle Text Loader

> **See Also:**
>
> *Oracle Database Utilities*

7.1 Basics of Oracle Database Utilities

The BS2000-specific information about Oracle Database utilities that you must use with Oracle Database 19c for Fujitsu BS2000 are discussed in the following sections:

- The Oracle Database Environment-Definition File
- Oracle Runtime Libraries
- Starting Oracle Utilities in the BS2000 Program Environment
- Starting Oracle Utilities in the POSIX Program Environment
- Connecting to an Oracle Database Instance
- Using BS2000 Files for Input and Output

7.1.1 The Oracle Database Environment-Definition File

Every Oracle Database utility and product under BS2000 uses an Oracle Database environment definition file, named `ORAENV`.

The `ORAENV` file is divided into the following parts:

- An executable part for BS2000 commands
- A static part for the definition of the environment variables

When processing the environment-definition file, a file link with the name ORAENV is created to the file itself. Oracle programs use the link name ORAENV to open this file. When reading this file, all lines with BS2000 commands ('/' in column one) and comments ('*' in column one) will be ignored. Only the variable settings starting in column one will be accepted.

You must generate this file before you use the Oracle Database programs as it contains several Oracle Database environment variables. These Oracle Database environment variables describe the operating environment for the Oracle database and utilities.

If you do not generate the ORAENV file, then the default values are used for all environment variables. In some cases, there are no default values for environment variables, such as for ORASID. If you start an Oracle Database program or utility without generating the ORAENV file first, then you cannot connect to the Oracle Database.

This section includes the following topics:

- Generating the Environment-Definition File
- Calling the Environment Definition File
- Specifying the Environment Variables

7.1.1.1 Generating the Environment-Definition File

To generate an ORAENV file, perform the following steps:

1. Call the INSTALL.P.USER procedure by entering the following command:

   ```
   /CALL-PROCEDURE $ORACINST.INSTALL.P.USER
   ```

   You are prompted to enter the database system identifier, *SID*.

2. Enter the *SID*. If you do not know the *SID*, then contact your database administrator.

7.1.1.2 Calling the Environment Definition File

Use the BS2000 command CALL-PROCEDURE to process the ORAENV file. For example:

```
/CALL-PROCEDURE DEMO.P.ORAENV
```

7.1.1.3 Specifying the Environment Variables

To specify the environment variables, call the ORAENV file containing the environment variables for the database you want to use.

If required, you can change the Oracle Database 19c working environment by editing the user variables in the ORAENV file.

> **Note:**
>
> The values that you assign to user variables are specific to your task and the database with which you work. The database administrator can also set other variables that may affect the whole database instance. If you try to set values for the DBA-specific variables in the `ORAENV` file, then they are ignored.

> **See Also:**
>
> Oracle Environment Variables for a list of the variables that you can specify in the `ORAENV` file

7.1.2 Oracle Runtime Libraries

The executables of Oracle Database 19c are stored in a library called `ORALOAD.LIB`. Before running an Oracle program you must assign this library to the link name `ORALOAD`. This link is created when the `ORAENV` procedure is called. If the link `ORALOAD` is not defined properly, then a BLS (BS2000 loader) error message is displayed.

The `ORAMESG` library, `$ORACINST.ORAMESG.LIB` is required for Oracle messages. This library is assigned the link name `ORAMESG` in the `ORAENV` procedure.

7.1.3 Starting Oracle Utilities in the BS2000 Program Environment

Before you start Oracle Database programs, you must call the environment definition file.

> **See Also:**
>
> "Calling the Environment Definition File" for more information

Use the SDF command `START-EXECUTABLE-PROGRAM` or the shorter form `START-EXECUTABLE` to start a program or a utility. Specify the options and operands as the first data input line when the data prompt (*) is displayed, as shown in the following example:

```
/START-EXECUTABLE (*LINK(ORALOAD),program_name)
CCM0001 enter options:
* [option_switch] [arguments]
```

where:

`program_name` is the name of the program or the utility that you want to start

`option_switch` is one or more of the program-dependent optional switches. If this is used, then the switch is preceded by a dash (-).

`arguments` are one or more operands of the program (or utility), or the user name and password combination, or both.

As soon as the program is loaded, the CCM0001 prompt is displayed and enables you to enter the command line options. As shown in the preceding examples, you can enter the *option_switch* or *arguments* for the program. Then the prompt of the program is displayed. If the program is SQL*Plus, then the prompt is SQL>. You can now enter one of the commands of the program. See the generic documentation for the utility for a description of the valid commands.

For example, to start SQL*Plus, enter the following command:

```
/START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
* userid/password
```

To start a utility in UNIX-Style, the Oracle syntax file $ORACINST.SYSSDF.ORACLE.USER must be activated. This is done by the MOD-SDF (MODIFY-SDF-OPTIONS) command in the ORAENV file. Remove the comment marker '&*' and call the *sid*.P.ORAENV file again. The following commands to start an Oracle utility are available:

```
/START-ORACLE-CM-CONTROL         or     /CMCTL
/START-ORACLE-CMMIGR             or     /CMMIGR
/START-ORACLE-EXPORT             or     /OEXP
/START-ORACLE-EXPDP              or     /EXPDP
/START-ORACLE-IMPORT             or     /OIMP
/START-ORACLE-IMPDP              or     /IMPDP
/START-ORACLE-LISTENER-CONTROL   or     /LSNRCTL
/START-ORACLE-ORAPWD             or     /ORAPWD
/START-ORACLE-SQLLOADER          or     /SQLLDR
/START-ORACLE-SQLPLUS            or     /SQLPLUS
/START-ORACLE-TNSPING            or     /TNSPING
/START-ORACLE-RECOVERY-MANAGER   or     /RMAN
```

Specify the parameters after the start command. Enclose the parameters within single quotation marks, if they contain a white space or an equal sign (=) . The following examples show how to start an utility in UNIX Style in the BS2000 program environment:

```
/lsnrctl

/sqlplus /nolog

/oimp 'system/manager file=iea buffer=210000 ignore=y grants=y rows=y
full=y commit=y'
```

7.1.4 Starting Oracle Utilities in the POSIX Program Environment

You can run utilities like SQL*Plus both in the normal BS2000 environment and the POSIX environment.

During Oracle Database installation, the utilities are installed within the POSIX file system in the *oracle_home_path*/bin directory.

Before you start Oracle utilities in the POSIX shell, you must set the ORACLE_HOME environment variable and extend the PATH environment variable by the path name of the Oracle directory, *oracle_home_path*/bin as follows:

```
$ ORACLE_HOME=/u01/app/oracbase/product/19.3.0/dbhome_1
$ export ORACLE_HOME
```

```
$ PATH=$ORACLE_HOME/bin:$PATH
$ export PATH
```

Alternatively, you can execute the *oracle_home_path*/.profile.oracle profile, which is created during Oracle Database installation under POSIX. This profile sets and expands the most important variables like ORACLE_HOME and PATH.

To execute the profile, enter the following command:

```
$ . /u01/app/oracbase/product/19.3.0/dbhome_1/.profile.oracle
```

Set the variable ORACLE_SID before you start an Oracle utility for a specific Oracle Database instance. Check the related ORAENV file in the BS2000 file system for instance specific environment variables. Set these variables in the POSIX shell before you start the utility.

Utilities running in the POSIX shell provide the opportunity to read instance-specific variables from the ORAENV file in the BS2000 file system. To provide access to the BS2000 ORAENV file, you must create a file with the name, oraenv*sid* in the ORACLE_HOME/dbs directory or ORACLE_BASE_CONFIG/dbs directory in a read-only Oracle home environment. This file contains the fully qualified BS2000 file name of the BS2000 ORAENV file. It acts as a link to the ORAENV file in the BS2000 file system.

For example, to access the ORAENV file, $ORADATA.ORCL.P.ORAENV, you must create an oraenvORCL file in the ORACLE_HOME/dbs directory, as follows:

```
$ ORACLE_HOME=/u01/app/oracbase/product/19.3.0/dbhome_1
$ export ORACLE_HOME
$ echo '$ORADATA.ORCL.P.ORAENV' > $ORACLE_HOME/dbs/oraenvORCL
$ chmod 664 $ORACLE_HOME/dbs/oraenvORCL
```

Note:

- Utilities running in the POSIX shell handle the variables of the BS2000 ORAENV file as subordinate variables. Environment variables in the POSIX shell take precedence over settings in the BS2000 ORAENV file.
- The *sid* in the file name oraenv*sid* is case sensitive and must match the *sid* specified in ORACLE_SID.
- You must grant access to the user using the BS2000 ORAENV file, if the POSIX user that runs the Oracle utility in the POSIX shell is different from the BS2000 user ID where the ORAENV file is located.

If an Oracle utility uses the BEQ protocol to connect to a database, then Oracle Net Services gets the job parameters to start a dedicated server in the BS2000 environment from the BGJPAR variable. If you do not specify this variable, then Oracle Net Services uses default values.

> **Note:**
>
> The BGJPAR variable might not be set after the *oracle_home_path*/.profile.oracle profile is run.

When using the BEQ protocol, Oracle recommends that you define the particular BS2000 job parameters for BS2000 jobs that are started by Oracle Net Services. The BGJPAR variable provides the option to define these parameters. You can define this variable either in the related BS2000 ORAENV file or by explicitly setting it in the POSIX environment to the appropriate value.

For example, to assign a bequeathed server task to a special JOB-CLASS, set the BGJPAR variable in the POSIX environment as follows:

```
$ ORACLE_SID=orcl
$ export ORACLE_SID
$ BGJPAR='START=SOON,CPU-LIMIT=NO,JOB-CLASS=JCBORA,LOGGING=*NO'
$ export BGJPAR
```

You can start the utilities in a similar way as on other UNIX systems. For example, to start SQL*Plus, use the following commands:

```
$ sqlplus /nolog
$ SQL> connect / as sysdba
```

7.1.5 Connecting to an Oracle Database Instance

You can connect to an Oracle Database instance using one of the following methods:

- Oracle Net Services with the Bequeath adapter.
- Oracle Net Services over TCP/IP or IPC.

Check with your database administrator if you can connect to the Oracle database using the listed methods, as the possibilities available are dependent on how the system has been configured. Usually, you specify the way you connect to an Oracle Database instance as part of the logon string appended to the *userid/password*, and separated from it by an at sign (@), as described in the following sections:

Related Topics

- Default Connections
- Accessing an Oracle Database Instance

7.1.5.1 Default Connections

If you do not specify a connection string, then use the DEFAULT_CONNECTION or TWO_TASK environment variables to specify an Oracle Database Net Services connect descriptor.

> **See Also:**
>
> "Oracle Environment Variables" for more information about the ORAENV file, and the DEFAULT_CONNECTION and TWO_TASK environment variables.

7.1.5.2 Accessing an Oracle Database Instance

Use Oracle Net Services to access a local or a remote database instance. Use the Oracle Net Services logon string to identify the following connection attributes for accessing a local or a remote database:

- Protocol to be used.
- Database that you want to access.
- Type of server (whether dedicated or shared) that you want to use.

The Oracle Net Services logon string has the following structure:

```
/START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
* userid/password@net_service_name
```

where:

net_service_name specifies a service name stored in the TNSNAMES.ORA file that identifies the TNS connect descriptor for the desired database. If you are not sure of what you should enter, then contact your database administrator.

The following example shows a logon string to connect to a database defined in the TNSNAMES.ORA file as SERVERX:

```
username-for-HR/password-for-HR@SERVERX
```

> **See Also:**
>
> "Oracle Net Services" for information about connecting to an Oracle Database using the Bequeath adapter

7.1.6 Using BS2000 Files for Input and Output

In most cases, Oracle Database for BS2000 programs use the functions of the C-BS2000 run-time system to access their input and output files. Oracle Database programs can read and write SAM, ISAM, and PAM files.

This section includes the following topics:

- Text Files
- Binary Files
- Default File Name Extensions
- Using Link Names
- Fixed Link Names

7.1.6.1 Text Files

SAM or ISAM files store textual data. Each record is considered as a single text line. For example, the SQL script files used by SQL*Plus and spool output files.

SQL*Loader input data is provided as SAM or ISAM files. These files may also contain non-printable data, such as packed decimal or binary integer values. For ISAM files, the key at the beginning of the record is generally ignored.

7.1.6.2 Binary Files

Binary data is usually stored in PAM files.

7.1.6.3 Default File Name Extensions

Under BS2000, the Oracle Database utilities add default extensions to file names only if the last component of the specified file name is longer than three characters, or if only one component is specified, as shown in the following table:

Original File Name	Extended File Name
TEST.TEST	TEST.TEST.EXT
TST	TST.EXT
T.T	T.T
TEST.TST	TEST.TST

This is similar to the file naming conventions used with Oracle Database on a UNIX system.

7.1.6.4 Using Link Names

In special cases, instead of specifying a file name, you can also specify the link name of a previously issued BS2000 /SET-FILE-LINK command. Use the syntax link=*linkname* in places where a file name is requested. In this way, you can override default file attributes, preallocate file space, and so on. There are a few exceptions where you cannot use the link=*linkname* notation.

When using the link=*linkname* notation, the default file name extensions do not work. As a result, file name defaults derived from such notation are not valid, and you must provide explicit names in such cases.

For example, when working with SQL*Loader, if you specify link=*linkname* for the SQL*Loader control file, then you must provide explicit names for the BAD, LOG, and DISCARD file names.

Some programs may report a syntax error when the link=*linkname* notation is used on the command (options) line. In such cases, omit the parameter on the command line and specify it instead, when you are prompted for the missing parameter.

7.1.6.5 Fixed Link Names

Oracle Database for BS2000 uses fixed link names for specific files.

The most important link names are as follows:

Type	Meaning/Usage
ORAENV	Oracle Database environment definition file.
ORALOAD	The link name is mandatory and specifies the load library from which the Oracle Database modules are loaded during processing.
ORAMESG	The link name is mandatory and specifies the message library from which Oracle message modules are loaded during execution.

Typically, you can set these link names by running the `ORAENV` procedure.

7.2 SQL*Plus

SQL*Plus is an interactive and a batch query tool that is installed with every Oracle Database installation. It has a command-line user interface. SQL*Plus has its own commands and environment, and provides access to the Oracle Database. It enables you to enter and execute SQL, PL/SQL, SQL*Plus, and operating system commands to perform the following:

- Format, perform calculations on, store, and print from query results
- Examine table and object definitions
- Develop and run batch scripts
- Perform database administration

This section describes how to use SQL*Plus in the BS2000 environment and in the POSIX environment. It supplements the *SQL*Plus User's Guide and Reference*. It contains the following topics:

- Using SQL*Plus in the BS2000 Environment
- Using SQL*Plus in the POSIX environment
- SQL*Plus User Profiles
- Using SQL*Plus Symbols
- Sample Schemas and SQL*Plus
- SQL*Plus Limits

7.2.1 Using SQL*Plus in the BS2000 Environment

This section describes how to use SQL*Plus in the BS2000 environment. It contains the following topics:

- Starting SQL*Plus in the BS2000 Environment
- Interrupting a SQL*Plus Command in the BS2000 Environment
- Running BS2000 Commands from SQL*Plus
- Starting the BS2000 Editor
- Spooling SQL*Plus Output
- Specifying the Search Path for SQL Scripts in the BS2000 Environment
- Starting SQL*Plus in a BS2000 command procedure

7.2.1.1 Starting SQL*Plus in the BS2000 Environment

1. To start SQL*Plus, enter:

   ```
   /START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
   *username/password
   ```

   > **Note:**
   >
   > If you omit user name and password, then you will be prompted to enter these values.
   >
   > If you enter the user name only, then you will be prompted for the password.

   SQL*Plus displays the following command prompt:
   `SQL>`

   The SQL*Plus command prompt indicates that SQL*Plus is ready to accept your commands.

2. To start SQL*Plus without connecting to a database, enter the following command:

   ```
   /START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
   */NOLOG
   ```

3. After SQL*Plus has started, you can connect to the database with the `CONNECT` command as follows:

   ```
   SQL> CONNECT username/password
   ```

7.2.1.2 Interrupting a SQL*Plus Command in the BS2000 Environment

Use the INTERRUPT key [K2] to interrupt the execution of a command in SQL*Plus. For example, you can interrupt SQL*Plus if you receive a long report that you do not want to be completely displayed on the screen. When you press the INTERRUPT key [K2], the display of the report is stopped and the SQL*Plus command prompt is displayed.

> **Note:**
>
> If you issue an INTERRUPT key when an input is requested, then you must answer this request before interrupting the process. However, this answer will be ignored.

7.2.1.3 Running BS2000 Commands from SQL*Plus

The SQL*Plus `HOST` command and the `$` command enables you to run a BS2000 command without exiting from SQL*Plus.

Some examples of how you can use the HOST command:

- If you enter the HOST command without any BS2000 command, then it takes you to the BS2000 command level:

  ```
  SQL> HOST
  /SHOW-USER-STATUS
  ```

  To return to SQL*Plus, use the BS2000 command RESUME .
- If you enter the HOST command with a BS2000 command, then the command executes and the control returns to the SQL*Plus command level:

  ```
  SQL> HOST SHOW-USER-STATUS
  SQL>
  ```

The following BS2000 commands, if used with the HOST or $ command, do not return to SQL*Plus when they have finished running:

- START-EXECUTABLE-PROGRAM
- LOAD-EXECUTABLE-PROGRAM
- START-PROGRAM
- LOAD-PROGRAM
- CALL-PROCEDURE
- HELP-SDF
- LOGOFF

7.2.1.4 Starting the BS2000 Editor

You can use the SQL*Plus EDIT command to start the BS2000 editor:

```
SQL> EDIT
```

This command:

- Writes the SQL buffer, which contains the current SQL statement, to a file called SQLEDT.BUF in the current BS2000 user ID.
- Starts the editor EDT, which reads the SQLEDT.BUF file into the work area.

You can then edit and write to the file using the @write command. Use the @halt command to exit the editor and return to SQL*Plus. SQL*Plus then reads the current contents of SQLEDT.BUF back into its command buffer, from which you can execute the SQL statement by entering a forward slash (/) at the SQL*Plus command prompt.

> **Note:**
>
> If you use the SQL*Plus DEFINE _EDITOR command to define a name for the editor, then SQL*Plus ignores it when running in the BS2000 environment. It always starts the EDT editor.

You can also use the EDIT command to edit a SQL file by specifying the SQL file. For example, if you enter the following command, then the editor EDT is called to edit the LOGIN.SQL file:

```
SQL> EDIT LOGIN[.SQL]
```

Note that you can omit the default file name extension .SQL.

> **See Also:**
>
> *SQL*Plus User's Guide and Reference* for more details about the EDIT command

7.2.1.5 Spooling SQL*Plus Output

When you use the SQL*Plus SPOOL command, SQL*Plus uses the default output-file suffix, .LST.

> **Note:**
>
> The output generated by BS2000 operating system commands are not spooled.

When you issue a SPOOL OUT command, SQL*Plus executes the BS2000 /PRINT command:

```
/PRINT tempfile,ERASE
```

where tempfile is a temporary copy of the spool file. This routes the file to the central printer. To specify any /PRINT command options, such as character sets, or routing to a remote printer, add the following line to the ORAENV file:

```
PRINTPAR=options
```

where options is any sequence of /PRINT command options. SQL*Plus then executes a /PRINT command, which includes these options.

> **See Also:**
>
> BS2000 manual *User Commands (ISP Format)* for more information about these options

7.2.1.6 Specifying the Search Path for SQL Scripts in the BS2000 Environment

You can run SQL script with SQL*Plus by using the START command or the @ (at symbol).

If SQL*Plus is executed in the BS2000 environment, it searches the SQL script in the current BS2000 user ID. If the script is not found, then SQL*Plus searches the paths specified by the SQLPATH environment variable. This variable is used to specify one or

more file name prefixes separated by a semicolon (;), which should be applied when searching for the SQL script file.

For example, if SQLPATH is set as follows:

```
SQLPATH=PRIVATE;$GLOBAL;/guest/scripts/;
```

then, when you enter the following command:

```
SQL> @filename
```

SQL*Plus searches for the SQL script file in the following sequence, until a matching file name is found:

1. *filename*.SQL in the current BS2000 user ID in the BS2000 DMS.
2. PRIVATE.*filename*.SQL with the prefix PRIVATE in the current user ID in the BS2000 DMS.
3. $GLOBAL.*filename*.SQL in the BS2000 user ID $GLOBAL in the BS2000 DMS.
4. */guest/scripts/filename.sql* in the directory /guest/scripts/ in the POSIX file system.

7.2.1.7 Starting SQL*Plus in a BS2000 command procedure

If you execute SQL*Plus within a BS2000 SDF command procedure, add the following command to the procedure before the /START-EXECUTE command for SQL*Plus:

```
/ASSIGN-SYSDTA *SYSCMD
```

This forces SQL*Plus to read data from the procedure, instead of prompting at the terminal.

7.2.2 Using SQL*Plus in the POSIX environment

You can run SQL*Plus not only in the normal BS2000 environment, but also in the POSIX environment.

This section describes the following:

- Starting SQL*Plus in the POSIX Environment
- Interrupting a SQL*Plus Command in the POSIX Environment
- Running Shell Commands From SQL*Plus
- Using an Editor in SQL*Plus
- Spooling SQL*Plus Output in the POSIX Environment
- Specifying the Search Path for SQL Scripts in the POSIX Environment

7.2.2.1 Starting SQL*Plus in the POSIX Environment

Set the required environment variables and run SQL*Plus as described in "Starting Oracle Utilities in the POSIX Program Environment." To connect to a database immediately, specify the user name and password as an argument as follows:

```
$ sqlplus username/password[@net_service_name]
```

If you do not want to connect to a database, then specify */nolog* as an argument:

```
$ sqlplus /nolog
```

If you start SQL*Plus without any arguments, then you will be prompted for the user name and password.

7.2.2.2 Interrupting a SQL*Plus Command in the POSIX Environment

Interrupting a SQL statement when you run SQL*Plus in the POSIX environment depends on the terminal connected with your POSIX session.

To interrupt a SQL statement in the POSIX environment:

- Enter @@*c* and then press the **Enter** key if the POSIX shell is started on a blockmode terminal.
- Press the **Ctrl + C** key combination if the POSIX shell is started by a remote X-client through *rlogin* or *ssh* using an xterm terminal.

7.2.2.3 Running Shell Commands From SQL*Plus

The SQL*Plus `HOST` command and the `$` command enables you to enter a POSIX shell command, without exiting SQL*Plus.

When using the `HOST` command, remember the following points:

- If you enter the `HOST` command without any shell command, then a POSIX subshell is started and the POSIX command prompt is displayed. To return to SQL*Plus, you must use the `exit` command or the `return` command in the POSIX subshell.
- If you enter the `HOST` command with a POSIX shell command, then the command is executed in a subshell and then control returns to SQL*Plus.
- Use the `bs2cmd` POSIX shell command to execute BS2000 commands.

7.2.2.4 Using an Editor in SQL*Plus

Start a text editor in SQL*Plus with the `EDIT` command, to edit an SQL statement.

The default editor depends on the terminal connected with your POSIX session:

- If the POSIX shell is started on a blockmode terminal, then `edtu` is set as the default editor in SQL*Plus.
- If the POSIX shell is started by a remote X-client through `rlogin` or `ssh` using an xterm terminal, then `vi` is set as the default editor in SQL*Plus.

SQL*Plus provides the opportunity to define a preferred text editor with the `DEFINE _EDITOR` command. In the POSIX environment you can define a preferred editor. For example, to define the editor used by the `EDIT` command to be the POSIX editor `edtu`, enter the following command in SQL*Plus:

```
DEFINE _EDITOR = edtu
```

> **Note:**
>
> - The editor `vi` does not work on blockmode terminals.
> - The editor `edtu` does not work on xterm terminals.

The command `EDIT`:

- Writes the SQL buffer, which contains the current SQL statement to a file called `SQLEDT.BUF` in the current working directory in the POSIX file system.
- Starts the editor, which reads the file `SQLEDT.BUF` into the work area.

You can then edit and write this file using `@write` with `edtu` editor and using `:w` with `vi` editor.

To exit the editor and return to SQL*Plus, use the `@halt` command with `edtu` and `:q` with `vi`. SQL*Plus then reads the current contents of `SQLEDT.BUF` back into its command buffer, from which the SQL statement can be run by entering slash "`/`" at the SQL*Plus command prompt.

You can also use the `EDIT` command to edit a SQL file by specifying the name of the SQL file. For example, if you enter the following command, then the editor is called to edit the `login.sql` file:

```
SQL> EDIT login[.sql]
```

Note that you can omit the default file name extension, `.sql`.

7.2.2.5 Spooling SQL*Plus Output in the POSIX Environment

When using the SQL*Plus `SPOOL` command, SQL*Plus uses the default output-file suffix, `.lst .`

The command `SPOOL OUT` is not supported, when you use SQL*Plus in the POSIX environment.

7.2.2.6 Specifying the Search Path for SQL Scripts in the POSIX Environment

You can run a SQL script with SQL*Plus by using the `START` command or the `@` (*at symbol*).

If SQL*Plus is executed in the POSIX environment, it searches the SQL script in the current working directory. If the script is not found, then SQL*Plus searches the paths specified by the `SQLPATH` environment variable. This variable specifies one or more file name prefixes separated by a semicolon (;), which should be applied when searching for the SQL script file.

For example, if `SQLPATH` is set as follows:

```
SQLPATH='private;/BS2/$GLOBAL;/guest/scripts/;'
export SQLPATH
```

then, when you enter the following command:

```
SQL> @filename
```

SQL*Plus searches for the SQL script file in the following sequence, until a matching file name is found:

1. *filename*.SQL in the current working directory in the POSIX file system.
2. private/*filename*.SQL in the subdirectory private of the current working directory in the POSIX file system.
3. /BS2/$GLOBAL.*filename*.SQL in the BS2000 user ID $GLOBAL in the BS2000 DMS.
4. */guest/scripts/filename.sql* in the directory /guest/scripts/ in the POSIX file system.

7.2.3 SQL*Plus User Profiles

You can set up your SQL*Plus environment to use the same settings with each session. There are two operating system files to do this:

- The Site Profile file, glogin.sql for site wide settings.
- The User Profile file, login.sql for user specific settings.

When a user starts SQL*Plus, the glogin.sql file is executed first followed by the user's login.sql file. These profiles are discussed in detail in the following topics:

- The glogin.sql Global Setup File
- The login.sql User Setup File

7.2.3.1 The glogin.sql Global Setup File

The Site Profile file, glogin.sql is executed when you start SQL*Plus. This file contains SQL statements and SQL*Plus commands that must be run at the beginning of a SQL*Plus session. The glogin.sql file is located in the POSIX file system, $ORACLE_HOME/sqlplus/admin. The database administrator may customize the glogin.sql file if required.

7.2.3.2 The login.sql User Setup File

Every time you start SQL*Plus, the user profile script login.sql is executed after the glogin.sql script. Similar to glogin.sql, this file also contains SQL statements and SQL*Plus commands that a user wants to run at the beginning of every SQL*Plus session.

When you start SQL*Plus in the BS2000 environment, SQL*Plus searches along the path that is specified by the ORACLE_PATH environment variable. If there are more than one login.sql file, then SQL*Plus executes the first login.sql file that is found. If you specify a list of login.sql files, they must be separated by a semicolon (;). For a customized SQL*Plus environment, each BS2000 user ID can have its own login.sql file.

When you start SQL*Plus in the POSIX environment, SQL*Plus searches along the path that is specified by the ORACLE_PATH environment variable . If there are more than one login.sql file, then SQL*Plus executes the first login.sql file that is found. For a customized SQL*Plus environment, each POSIX user can have its own login.sql file.

> **See Also:**
>
> - "Oracle Environment Variables" for a description of the SQLPATH environment variable
> - *SQL*Plus User's Guide and Reference* for more information about `login.sql`

The following is a sample profile file:

```
set echo off
set feedback 4
set pause on
set pause  PLEASE ACKNOWLEDGE TO CONTINUE
set echo on
```

7.2.4 Using SQL*Plus Symbols

The SQL*Plus symbol used for concatenation is the vertical bar, "|" (X'4F'). For users with German keyboards and using a 7–bit terminal character set, any key that transmits a X'4F' (for example, "ö"), can be used.

7.2.5 Sample Schemas and SQL*Plus

The sample schemas provide a common platform for examples.

> **See Also:**
>
> - *SQL*Plus User's Guide and Reference* for more information about the sample schemas and SQL*Plus
> - "Creating a Database" for information about how to install the sample schemas

7.2.6 SQL*Plus Limits

The limits of several SQL*Plus elements are specified in *SQL*Plus User's Guide and Reference*. The following table defines BS2000 specific limits:

Item	Limit
File name length	54 in the BS2000 DMS (including CATID and BS2000 user ID) 512 in the POSIX file system
LINESIZE	32767
MAXDATA	32767
Maximum number of nested command files	12

7.3 The SQL*Loader

SQL*Loader is a tool used for moving data from an external file (or files) into the tables of an Oracle database. SQL*Loader can load data in several formats and can even load several tables simultaneously. You can also use it to load only records that match a particular data value.

This section includes the following topics:

- Starting the SQL*Loader Utility
- Using the SQL*Loader Demonstration Files

> **See Also:**
>
> *Oracle Database Utilities* for a detailed description of SQL*Loader and its demonstration files

7.3.1 Starting the SQL*Loader Utility

You can start SQL*Loader either in the BS2000 environment or the POSIX environment.

Related Topics

- Starting Oracle Utilities in the BS2000 Program Environment
- Starting Oracle Utilities in the POSIX Program Environment

7.3.2 Using the SQL*Loader Demonstration Files

The demonstration files are shipped under:

```
$ORACINST.RDBMS.DEMO.ULCASE*.CTL
$ORACINST.RDBMS.DEMO.ULCASE*.SQL
$ORACINST.RDBMS.DEMO.ULCASE*.DAT
```

To run the ULCASE1 demo, perform the following steps:

1. Run SQL*Plus and set up the table to be used in the demonstration by entering the following commands:

   ```
   /START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
   * SCOTT/password
   SQL> START $ORACINST.RDBMS.DEMO.ULCASE1
   ```

   > **Note:**
   >
   > This example sets up the table for the user `SCOTT` to run the demonstrations.

2. Start SQL*Loader to run the demonstration by entering the following command:

```
/START-EXECUTABLE (*LINK(ORALOAD),SQLLDR)
* SCOTT/password $ORACINST.RDBMS.DEMO.ULCASE1 ULCASE1 ULCASE1
```

7.4 The Export Utility

The Export utility is used to write data from an Oracle database into the BS2000 files. Use this utility with the Import utility to back up your data and to move data between Oracle databases.

This section includes the following topics:

- Starting the Export Utility
- Exporting to Foreign Systems

7.4.1 Starting the Export Utility

You can start the Export utility, `exp` either in the BS2000 environment or the POSIX environment.

> **See Also:**
>
> - Starting Oracle Utilities in the BS2000 Program Environment
> - Starting Oracle Utilities in the POSIX Program Environment

Export dump files, which are created in the BS2000 DMS by `EXP` usually have the file structure, `SAM`. You can override default output file specifications by running a `SET-FILE-LINK` command such as:

```
/SET-FILE-LINK LINK-NAME=explink,FILE-NAME=expfile,ACCESS-
METHOD=*SAM,RECORD-FORMAT=*FIXED,RECORD-SIZE=2048,BUFFER-LENGTH=*STD(1)
```

Then, call the `EXP` utility by specifying the following in response to the output file name prompt:

```
LINK=explink
```

On a nonkey public volume set, you may need to adjust the `BLKSIZE` and `RECSIZE` values for efficient disk-space usage (note that `RECSIZE` must be 16 bytes less than the `BLKSIZE` on nonkey disks). Specify the `RECSIZE` value to match the export record size.

For example:

```
/SET-FILE-LINK LINK-NAME=explink,FILE-NAME=expfile,ACCESS-
METHOD=*SAM,RECORD-FORMAT=*FIXED,RECORD-SIZE=2032,BUFFER-LENGTH=2048
```

> **Note:**
>
> Do not use variable record size with SAM files.
>
> When using a block size (PAM) or record size (SAM) other than 2048, you must also specify a corresponding RECORDLENGTH parameter to EXP on the options line.

When exporting large volumes of data, the default disk-space allocation for the output file is inappropriate, and the program spends a significant amount of time allocating secondary extents of disk space. If the maximum number of extents exceeds the number that the BS2000 catalog entry can hold, then an output-file error occurs.

You can preallocate the EXP output file with the BS2000 CREATE-FILE command, before starting the Export utility. When allocating the file, you must use a realistic estimate for both the primary and secondary space allocations.

For example:

```
/CREATE-FILE LARGE.EXPORT.DMP,SPACE=(30000,30000)
/ADD-FILE-LINK LINK-NAME=EXPOUT,FILE-NAME=LARGE.EXPORT.DMP
/START-EXECUTABLE (*LINK(ORALOAD),EXP)
* system/manager
...
Export file: EXPDAT.DMP >link=expout
...
```

7.4.2 Exporting to Foreign Systems

You can export to foreign systems using the following methods:

- Exporting Data to Tape
- Transferring Data by File Transfer

7.4.2.1 Exporting Data to Tape

To export directly to tape:

1. Create a catalog entry for a file using the CREATE-FILE command.
2. Create a link using the ADD-FILE-LINK command. For example:

   ```
   /CREATE-FILE tapefile,SUPPORT=*TAPE(VOLUME=vsn,DEVICE-TYPE=device)
   /ADD-FILE-LINK
   LINK-NAME=tapelink,FILE-NAME=tapefile,ACCESS-METHOD=*SAM,RECORD-
   FORMAT=*FIXED,RECORD-SIZE=2048,BUFFER-LENGTH=*STD(1)
   ```

3. Set the environment variable EXP_CLIB_FILE_IO to FALSE.
4. Execute EXP by specifying the following value in response to the output file name prompt:

   ```
   LINK=tapelink
   ```

The export utility writes the output as SAM files, which simplifies export to an Oracle database on foreign systems.

7.4.2.2 Transferring Data by File Transfer

If you use FTP, then ensure that you specify binary mode. This is to avoid automatic EBCDIC-ASCII conversion.

You must use FTP on BS2000 when you want to use an export file from BS2000 as an import file on an ASCII platform. To avoid new line (NL) insertion at block boundaries, ensure that you provide the `binary` and the `ftyp binary` parameters.

7.5 The Import Utility

The Import utility is used to write data from the files created by the Export utility to an Oracle database.

This section includes the following topics:

- Starting the Import Utility
- Importing from Foreign Systems

> **See Also:**
>
> "Known Problems, Restrictions and Workarounds" in *Oracle Database Release Notes for Fujitsu BS2000* for restrictions when using the Import utility

7.5.1 Starting the Import Utility

You can start the Import utility, `imp` either in the BS2000 environment or the POSIX environment.

> **See Also:**
>
> - Starting Oracle Utilities in the BS2000 Program Environment
> - Starting Oracle Utilities in the POSIX Program Environment

7.5.2 Importing from Foreign Systems

Follow the guidelines listed in this section when you import data from non-BS2000 systems:

- Importing File with Non-Standard Block Size
- Importing Data from Tape
- Transferring Data by File Transfer

7.5.2.1 Importing File with Non-Standard Block Size

If the import file on the BS2000 operating system has a block size (BLKSIZE) not equal to 2 KB, then you must specify the block size during import with the Import parameter RECORDLENGTH.

7.5.2.2 Importing Data from Tape

The Import utility can read directly from tape, provided the file can be processed as a SAM file, which is usually the case even for EXP files created on foreign systems (for example, as a sequence of fixed 2 KB blocks).

To read a foreign export file directly:

1. Create a catalog entry for a file using the IMPORT-FILE command.
2. Create a link using the ADD-FILE-LINK command. For example:

   ```
   /IMPORT-FILE SUPPORT=*TAPE(VOLUME=vsn,DEVICE-TYPE=device,FILE-
   NAME=tapefile)
   /ADD-FILE-LINK LINK-NAME=tapelink,FILE-NAME=tapefile
   ```

3. Set the environment variable IMP_CLIB_FILE_IO to FALSE.
4. Execute IMP by specifying the following value in response to the input file name prompt:

   ```
   LINK=tapelink
   ```

7.5.2.3 Transferring Data by File Transfer

If you use FTP, then ensure that you specify binary mode. This is to avoid automatic ASCII-EBCDIC conversion. The received file is stored as a PAM file by the BS2000 FTP utility and can immediately be used as an input file to IMP.

7.6 The Data Pump Export Utility

The Data Pump Export and Import are functionally similar to Export and Import discussed in the preceding sections. However, the I/O processing for dump files is done in the Oracle database server rather than in the client utility session.

This utility is used to write data from an Oracle database into the BS2000 files. Use this utility with the Data Pump Import utility to back up your data and to move data between Oracle databases.

7.6.1 Starting the Data Pump Export Utility

You can start the Data Pump Export utility, expdp either in the BS2000 environment or the POSIX environment.

> **See Also:**
>
> - "Starting Oracle Utilities in the BS2000 Program Environment"
> - "Starting Oracle Utilities in the POSIX Program Environment"

The Data Pump Export dump files are always created in the BS2000 DMS as `PAM` files with `BLKSIZE=(STD,2)`.

To use an export file from BS2000 as an import file on an ASCII platform, use FTP as the transfer utility on BS2000 side and indicate the `binary` parameter .

> **Note:**
>
> If you start `EXPDP` in UNIX-style and use the interactive-command mode [K2] key, then the parameters must be specified when you are prompted for them, and not on the command line.
>
> Data Pump Export to tape is not supported.

7.7 The Data Pump Import Utility

The Data Pump Import utility is used to write data from the files created by the Data Pump Export utility to an Oracle database.

7.7.1 Starting the Data Pump Import Utility

You can start the Data Pump Import utility `impdp` either in the BS2000 environment or the POSIX environment.

The following example shows how to use the command-line mode of `impdp` in the BS2000 environment:

```
START-EXECUTABLE (*LINK(ORALOAD),IMPDP)
* username/password [options]
```

If you use an export file from an ASCII platform as an import file on BS2000, then use FTP as the transfer utility on BS2000 side and indicate the `binary` parameter.

Before you get the file, issue the FTP command:

```
file dmp-file,fcbtype=pam,blksize=(std,2),blkctrl=no
```

> **Note:**
>
> If you start IMPDP in UNIX-style and use interactive-command mode [K2] key, then the parameters must be specified when you are prompted for them, and not on the command line.
>
> Data Pump Import by tape is not supported.

Related Topics

- Starting Oracle Utilities in the BS2000 Program Environment
- Starting Oracle Utilities in the POSIX Program Environment

7.8 Recovery Manager on BS2000

On BS2000, Recovery Manager does not support tapes. Disks are the only backup media.

As a workaround, use the Recovery Manager output as a first level storage to be migrated by BS2000 subsystem HSMS (Hierarchical Storage Management System) to tapes. However, it is the administrator's responsibility to ensure the cooperation of the two systems.

The following is an example of a Recovery Manager command in the BS2000 environment:

```
/START-EXECUTABLE (*LINK(ORALOAD),RMAN)
*target "dba1/dba1@i1" catalog "dba2/dba2@i2" cmdfile "b.dat" log "b.log"
```

7.9 Checking the Integrity of the Physical Data Structure

To check the physical data structure integrity of offline databases, use the DBVERIFY command-line utility. You can start the DBVERIFY utility dbv, either in the BS2000 environment or the POSIX environment.

The following examples shows how to use dbv in the BS2000 environment:

```
/START-EXECUTABLE (*LINK(ORALOAD),DBV)
file=orcl.dbs.database1.dbf blocksize=4096 feedback=100
```

The following example shows how to use dbv in the POSIX environment:

```
$ dbv file=/BS2/orcl.dbs.database1.dbf blocksize=4096 feedback=100
```

> **See Also:**
>
> - "Starting Oracle Utilities in the BS2000 Program Environment"
> - "Starting Oracle Utilities in the POSIX Program Environment"
> - *Oracle Database Utilities* for more information about the `DBVERIFY` program

7.10 Workload Replay Client

The Workload Replay Client (WRC) is a multithreaded program where each thread submits a workload from a captured session.

An executable program, `wrc` is located in the `$ORACLE_HOME/bin` directory. The specific characteristics of the Workload Replay Client (WRC) on BS2000 is discussed in the following topics:

- About Running Workload Replay Client
- About Troubleshooting Workload Replay Client

Related Topics

- *Oracle DatabaseTesting Guide*

7.10.1 About Running Workload Replay Client

On BS2000 the WRC is built as a PTHREADS application with default options which enable the application to run out of the box. The user can modify these options for its own needs. The PTHREADS options are stored in the `WRC.OPTFILE` file, which is created by the installation procedure with the most important options in the installation user ID.

For a proper execution of the WRC, the `WRC.OPTFILE` file must reside in the BS2000 filesystem of the user ID where you run the WRC. Oracle recommends that you copy the `WRC.OPTFILE` file to the BS2000 filesystem of the desired user ID. The WRC itself can only be started in the POSIX shell.

```
/COPY-FILE $ORACINST.WRC.OPTFILE,WRC.OPTFILE,SAME
```

Before you can run the WRC you must prepare your database for testing as described in the *Oracle Database Testing Guide*. The captured data must be stored in the POSIX file system. If your database is ready for a workload replay, then you can run the WRC.

Start a POSIX shell, run the profile `oracle_home_path/.profile.oracle`, set an `ORACLE_SID` and start the WRC as described in the *Oracle Database Testing Guide*.

For example:

```
/START-POSIX-SHELL
$ ./u01/app/oracle/product/19.3.0/dbhome_1/.profile.oracle
$ ORACLE_SID=orcl
```

```
$ export ORACLE_SID
$ wrc system/password@test mode=replay replaydir=./replay
```

See Also:

- *Oracle Database Testing Guide* to prepare your database for testing
- *Oracle Database Testing Guide* for more information about starting WRC

7.10.2 About Troubleshooting Workload Replay Client

You need not modify the parameters of the `WRC.OPTFILE` except when there are problems with the WRC application. The following table shows the parameters defined in the `WRC.OPTFILE`. Modify the parameter values carefully:

Parameter	Value	Description
APPLL	$ORACINST.ORALOAD.LIB	Application Load Library
SCHPO	*STD|FIFO|RR Default: *STD	Defines the thread scheduling policy: Standard, First In First Out, or Round Robin.
BUSYC	Integer 1…2147483647 Default: 100	Number of retries to get a lock.
MINRT	Integer 0…59 Default: 1	Minimum number of resource tasks.
MAXRT	Integer 0…59 Default: 1	Maximum number of resource tasks.
MSESZ	Integer 1…2147483647 Default: 262144	Number of bytes reserved for the main stack.
ILCS7	Integer 1…2147483647 Default: 262144	Default size of a stack for a thread.
SHAMS	Integer 1…2147483647 Default: 131072	Number of memory pages reserved for the application memory pool.

To get information about the running WRC, call the procedure `ITH-SHOW` from the `SYSPRC` library of the `PTHREADS` installation:

```
/CALL-PROCEDURE $TSOS.SYSPRC.PTHREADS.014(ITH-SHOW)
```

This utility produces an output similar to the following:

```
/CALL-PROCEDURE $TSOS.SYSPRC.PTHREADS.014(ITH-SHOW)
%  BLS0523 ELEMENT 'ITHSHOW', VERSION 'V01.4A10', TYPE 'L' FROM LIBRARY
':BUG2:$
TSOS.SYSLNK.PTHREADS.014' IN PROCESS
%  BLS0524 LLM 'ITHSHOW', VERSION 'V01.4A10' OF '2013-11-30 03:17:44'
LOADED
%  BLS0551 COPYRIGHT (C) 2013 Fujitsu Technology Solutions. ALL RIGHTS
```

```
RESERVED
STARTED AT 2016-04-20-145816 BY POSIX  (running)
LLM     = WRC  (prelinked)
MAIN    = IC@#MAIN
APPLL   = :POR5:$ORA12102.ORALOAD.LIB
RUNTL   = :BUG2:$TSOS.SYSLNK.PTHREADS.014
PTHvers = 01.4A21 2016-02-10 18:55:31
FDs     = 5 (3 ORIG FDs, 2 RESO FDs)
Threads = 5 (1 user threads, 4 system threads)
TYPE  TSN   PID     JOB-TYPE           PRI      CPU-USED CPU-MAX
ACCOUNT#
ORIG  1D4U  2902 (X'0B56')  3 DIALOG  *0 240     3.0365   32767    FSC
                    waiting for new requests
RESO  1D48  2917 (X'0B65')  3 DIALOG  *0 240     0.0450   32767    FSC
                     executing request
THRE  1D49  2918 (X'0B66')  3 DIALOG  *0 240     0.2478   32767    FSC
```

Here you find the TSNs of the tasks involved in the `WRC` application. You can connect to the `WRC` application when you choose the TSN of the `ORIG` task as the input for the parameter TSN of the `ITH-START` procedure in the following format:

```
/CALL-PROCEDURE $TSOS.SYSPRC.PTHREADS.014(ITH-START),(TSN=1D4U)
```

When you see the double slash prompt you can type `SHOW-PTHREADS-STATUS` to show the status of the running `WRC` application or `CANCEL-THREADED-PROGRAM` to cancel the application. If the `ORIG` task is already terminated, then you can terminate all other tasks of the `WRC` application. In the BS2000 environment, use the following system command:

```
/CANCEL-JOB JOB-IDENTIFICATION=tsn
```

In the POSIX environment use the shell command `kill` to abort pending processes. You must use the signal `SIGABRT (6)` or `SIGTERM (15)` to notify the target process:

```
$ kill -6 pid
```

7.11 The Oracle Text Loader

This utility imports and exports text data. .

You can start the Oracle Text Loader utility `ctxldr` either in the BS2000 environment or the POSIX environment.

> **See Also:**
>
> Oracle Text for installing Oracle Text

The following examples show how to use `ctxldr` in the BS2000 environment:

```
/START-EXECUTABLE (*LINK(ORALOAD),CTXLDR)
*-USER username/password [options]
```

The following example shows how to use `ctxldr` in the POSIX environment:

```
$ ctxldr -USER username/password [options]
```

Related Topics

- *Oracle Text Reference*
- Starting Oracle Utilities in the BS2000 Program Environment
- Starting Oracle Utilities in the POSIX Program Environment

8
Backing Up and Recovering a Database

This chapter supplements the generic Oracle Database documentation set with information about backup and recovery. It includes the following topics:

- Backing Up an Oracle Database
- Restoring an Oracle Database
- About Using the Recovery Manager

Refer to the following Oracle manuals for detailed information about database backup and recovery:

- *Oracle Database Concepts*
- *Oracle Database Administrator's Guide*
- *Oracle Database Backup and Recovery User's Guide*

You can choose among many methods and Oracle tools for backup and recovery. You may use the Import and Export Utilities for logical backup and recovery. For physical backup and recovery you may use Recovery Manager (RMAN) or operating system utilities.

This chapter describes some BS2000 specific issues if you apply user-managed backup and recovery with SQL*Plus and BS2000 utilities.

8.1 Backing Up an Oracle Database

You can use one of the following methods to back up an Oracle Database:

- Using BS2000 Utilities to Back Up an Oracle Database
- Performing Online Backup

8.1.1 Using BS2000 Utilities to Back Up an Oracle Database

You can back up an Oracle database using BS2000 operating system utilities (for example, `ARCHIVE` or the `/COPY-FILE` command).

Use the following steps to back up an Oracle database:

1. While the database is running, collect the names of all the files, which make up the Oracle database. You can determine the names of the log and database files by entering the following commands:

```
/START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
* /NOLOG
SQL> CONNECT / AS SYSDBA
SQL> SELECT * FROM V$DATAFILE;
SQL> SELECT * FROM V$LOGFILE;
```

2. To ensure that all Oracle database files are synchronized at the time of the backup, shut down Oracle database using SQL*Plus.

3. Back up all the database files and log files using the BS2000 ARCHIVE utility or the BS2000 /COPY-FILE command. You should always back up all the files at the same time.
4. Restart Oracle Database using SQL*Plus.

8.1.2 Performing Online Backup

You can perform an online backup of the database or individual tablespaces by using either:

- The BS2000 ARCHIVE utility together with Oracle Database INSTALL.C.OPNBACK utility.
- The BS2000 PERCON utility.

The ARCHIVE method is faster, and is described in this section.

Before you can perform an online (hot) backup of individual tablespaces, you must ensure that the ARCHIVE utility can back up open files.

The following BS2000 command ensures that ARCHIVE can back up open files:

```
/START-EXECUTABLE $ORACINST.INSTALL.C.OPNBACK
*filename
```

The INSTALL.C.OPNBACK utility calls the BS2000 macro CATAL, which sets the OPNBACK file attribute to YES. For the CATAL macro to work, the database must be shut down or the tablespace in question must be offline. You enter this command once for each file. For example, before adding it to a tablespace, not on the occasion of each backup.

You must never back up database files online without first setting the tablespace to backup mode. If you do not follow this step, then the resulting backup files are inconsistent. To perform an online backup of individual tablespaces or data files, use the following procedure:

1. Enter the following command:

   ```
   SQL> ALTER TABLESPACE name BEGIN BACKUP;
   ```

2. Back up the files of the tablespace using the BS2000 ARCHIVE utility. Ensure that the OLS parameter of ARCHIVE is set to YES.
3. Enter the following command:

   ```
   SQL> ALTER TABLESPACE name END BACKUP;
   ```

> **Note:**
>
> The preceding SQL*Plus commands operate on tablespaces, while the ARCHIVE utility operates on data files.

8.2 Restoring an Oracle Database

An Oracle Database can be restored offline from backups, using the following steps:

1. Copy all the database files and the log files from the backup. You may use the BS2000 `ARCHIVE` utility or the BS2000 `/COPY-FILE` command. Files must be restored with their original name.

   To determine the name of all data files, query the `V$DATAFILE` table while the Oracle database is running. Enter the following command when the SQL prompt is displayed:

   ```
   SQL> SELECT FILE#,NAME FROM V$DATAFILE;
   ```

   The following is an example of the result that is displayed:

   ```
   FILE#    NAME
   ------    ------------------------
   1        :pvs:$dbauserid.sid.DBS.DATABASE1.DBF
   2        :pvs:$dbauserid.sid.DBS.DATABASE2.DBF

   2 ROWS SELECTED.
   ```

   You can determine the name of the log files in a similar way:

   ```
   SQL> SELECT GROUP#,MEMBER FROM V$LOGFILE;
   ```

   The following is an example of the result that is displayed:

   ```
   GROUP#    MEMBER
   ------    ------------------------
   1         :pvs:$dbauserid.sid.DBS.LOG1.DBF
   2         :pvs:$dbauserid.sid.DBS.LOG2.DBF

   2 ROWS SELECTED.
   ```

2. Under the DBA user ID, ensure that the `ORASID` environment variable identifies the Oracle database, which is to be restored.
3. Use the SQL*Plus `STARTUP` command to start the Oracle database.

8.3 About Using the Recovery Manager

In addition to the BS2000 utilities, you can also use Oracle Recovery Manager (RMAN) to back up and restore a database.

> **See Also:**
>
> "Recovery Manager on BS2000" for more information

9
About Unified Auditing

Unified auditing enables you to capture audit records from a variety of sources.

The unified audit trail, which resides in a read-only table in the AUDSYS schema in the SYSAUX tablespace, makes this information available in an uniform format in the UNIFIED_AUDIT_TRAIL data dictionary view.

When the database is writable, audit records are written to the unified audit trail. If the database is not writable, then audit records are written to new format operating system files in the POSIX file system in the $ORACLE_BASE/audit/$ORACLE_SID directory.

This chapter contains the following topics:

- Enabling Unified Auditing
- Disabling Unified Auditing

> **See Also:**
>
> *Oracle Database Security Guide* for more details

9.1 Enabling Unified Auditing

The Unified Auditing option is not enabled after you install Oracle Database. You can find if your database has been migrated to unified auditing by querying the V$OPTION dynamic view. Query the VALUE column of the V$OPTION dynamic view as follows with SQL*Plus:

```
SQL> SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Unified Auditing';
```

If the output for the VALUE column is TRUE, then pure unified auditing is already enabled in your database. If unified auditing has not been enabled, then the output is FALSE.

To enable the Unified Auditing option, relink the ORAKNL binary in the ORALOAD library $ORACINST.ORALOAD.LIB. Relinking is done by calling a BS2000 command procedure.

After shutting down all databases and stopping all listeners, log in to the installation user ID $ORACINST. Enter the following BS2000 command to enable Unified Auditing:

```
/CALL-PROCEDURE INSTALL.P.UNIAUD-ON
```

Restart the databases and listeners in your DBA user IDs. After restarting, all the databases run with Unified Auditing.

9.2 Disabling Unified Auditing

To disable the Unified Auditing option, relink the `ORAKNL` binary in the `ORALOAD` library `$ORACINST.ORALOAD.LIB`. Relinking is done by calling a BS2000 command procedure.

After shutting down all databases and stopping all listeners, log in to the installation user ID `$ORACINST`. Enter the following BS2000 command to disable Unified Auditing:

```
/CALL-PROCEDURE INSTALL.P.UNIAUD-OFF
```

Restart the databases and listeners in your DBA user IDs. After restarting, all databases run without the Unified Auditing option.

10
Java in the Database

This chapter describes BS2000-specific features for Java in the database. This chapter includes:

- Installation of a Java Enabled Database
- Database character sets and Java Encodings
- Java Demonstration Files

> **See Also:**
>
> Oracle Java documentation set for more information

10.1 Installation of a Java Enabled Database

When you call `$ORACINST.INSTALL.P.SUPER` and set the `JAVA` parameter to `YES`, you get a database that meets the Java requirements.

When you enable Java in an existing Oracle database, you can use the Java related parts of this procedure as an example and modify it according to your needs. For example, you can increase `dbsize`, `shared_pool_size`, run `initjvm.sql`, and so on.

Where can files related to Java reside and how should they be encoded?

It is not absolutely straightforward where files used by Java must be stored and how they should be encoded. In general, files can reside either in the native BS2000 or in the POSIX file system. However, there are exceptions.

The following table gives an overview of the file types, location, default encoding, and encoding modifications for APIs or statements:

Statement or API	File type	Place	Default encoding	Encoding modification
`CREATE JAVA CLASS USING BFILE`	.class	BS2000 PAM file or POSIX	Binary	Not applicable
`CREATE JAVA RESOURCE USING BFILE`	.properties	BS2000 PAM file or POSIX	ascii	None, that is, there is no means to change the default encoding.
`CREATE JAVA SOURCE USING BFILE`	.java, .sqlj	BS2000 PAM file or POSIX	DB charset	None, that is, there is no means to change the default encoding.

Statement or API	File type	Place	Default encoding	Encoding modification
`CREATE JAVA SOURCE AS`	.sql	Part of statement	Session character set specified in `NLS_LANG`	`NLS_LANG`
`CALL DBMS_JAVA.LOADJAVA`	*, .jar, .zip	POSIX	DB charset	Option encoding in loadjava call
`java.io-package`	*	POSIX	DB charset	Depends on the classes used

You can create BS2000 PAM files in ASCII by transferring files with the File Transfer Protocol (FTP) from an ASCII platform to BS2000 in binary mode.

Related Topics

- *Oracle Database Java Developer's Guide*

10.2 Database character sets and Java Encodings

As far as I/O is concerned, the Oracle JAVAVM uses the database character set as the system property `file.encoding`. Therefore, the following Oracle Database on Fujitsu BS2000 database character sets have been added to the list of supported Java encodings:

```
WE8BS2000
WE8BS2000E
EE8BS2000
CE8BS2000
CL8BS2000
WE8BS2000L5
```

These encodings are not known to any other Java implementation.

The system property `file.encoding`, however, does not apply to Java property files. Property files always use the encoding 8859_1. The system property `file.encoding` is used when compiling a source file. You can change this default by either setting the encoding option of the `dbms_java.loadjava` procedure or by using the following procedure:

```
dbms_java.set_compiler_option('','encoding',...)
```

> **See Also:**
>
> *Oracle Database SQLJ Developer's Guide* for more details about property files

10.3 Java Demonstration Files

A simple Java demonstration program running in the server is shipped under:

```
$ORACINST.JAVAVM.DEMO.HELLO.SQL
```

An example with database connection using the server-side internal driver is shipped under:

```
$ORACINST.JAVAVM.DEMO.EMPLOYEE.JAVA
$ORACINST.JAVAVM.DEMO.CREATE.SQL
$ORACINST.JAVAVM.DEMO.RUN.SQL
```

11
Oracle Text

Oracle Text provides indexing, word and theme searching, and viewing capabilities for text in query applications and document classification applications.

This chapter describes how to install and run Oracle Text and the restrictions of this option on BS2000. It includes the following topics:

11.1 Installing Oracle Text

Oracle Text is not installed when you create a database. To install Oracle Text, you must complete the following steps:

1. Start SQL*Plus. To avoid being prompted for many overflow acknowledgements on the screen, set `OVERFLOW-CONTROL=*NO-CONTROL`:

   ```
   /MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=*NO-CONTROL
   /START-EXECUTABLE (*LINK(ORALOAD),SQLPLUS)
   * /nolog
   connect / as sysdba
   spool catctx.log
   @$ORACLE_HOME/ctx/admin/catctx.sql CTXSYS SYSAUX TEMP NOLOCK;
   ```

   where `CTXSYS` is the password for `ctxsys`, `SYSAUX` is the default tablespace for `ctxsys`, `TEMP` is the temporary tablespace for `ctxsys`, and `LOCK|NOLOCK` specifies whether the `ctxsys` user account is locked or not.

2. If you are working with US english texts, then install the appropriate language-specific default preferences:

   ```
   connect CTXSYS/CTXSYS
   @$ORACLE_HOME/ctx/admin/defaults/drdefus.sql;
   ```

   If you are not working with US english texts, then open the `drdef*.sql` script in the `$ORACLE_HOME/ctx/admin/defaults` directory according to the preferred language, set the attributes, and run the script.

   Before you set the attributes, refer to "Restrictions of Oracle Text on BS2000."

3. Type `exit` when finished.

11.2 Restrictions of Oracle Text on BS2000

The following restrictions apply for Oracle Text on Fujitsu BS2000:

- Index themes are not supported.
- INSO filters, which are licensed on special platforms only, are not supported.
- `ctxkbtc`, which is a knowledge base utility is not supported.

- `URL_DATASTORE` objects are not supported.

> **Note:**
>
> `FILE_DATASTORE` objects may reside on native BS2000 DMS as PAM files or on the POSIX file system.

12
XML

This chapter describes BS2000 specific topics of XML such as installation, features, and restrictions. It contains the following topics:

- About XDK Installation
- Features and Restrictions of XML Features on BS2000 Systems

> **See Also:**
>
> - *Oracle XML Developer's Kit Programmer's Guide*
> - *Oracle XML DB Developer's Guide*
> - *Oracle Database XML C API Reference*
> - *Oracle Database XML C++ API Reference*
> - *Oracle Database XML Java API Reference*

12.1 About XDK Installation

The Jar files for the XML SQL utility, `xsu12.jar` and `xdb.jar`, and the XML Parser, `xmlparserv2.jar` are already loaded in the database when you create a database as explained in "About Creating a Database."

12.2 Features and Restrictions of XML Features on BS2000 Systems

The following table provides an overview of the XML features that are available for the programming languages on BS2000:

An empty field means that the feature is not supported.

N/A means means that it is not applicable.

XML Feature	Availability for Java	Availability for C	Availability for C++	Availability for PL/SQL
Parser	Yes			Yes
XSLT Processor	Yes			Yes
Class Generator		N/A		N/A
XSQL		N/A	N/A	N/A
Transviewer Beans		N/A	N/A	N/A

XML Feature	Availability for Java	Availability for C	Availability for C++	Availability for PL/SQL
XML-SQL Utility	Yes	N/A	N/A	Yes
Schema Processor	Yes			N/A

When you use PL/SQL instead of Java, you must consider the following behavior:

- PL/SQL file input is only possible from POSIX and with ASCII data format.
- PL/SQL file output is written to POSIX with ASCII data format.
- For `INSERT/UPDATE/DELETE` operations the XML document must not contain <? `xml ... encoding=WE8BS2000 ...>`.

When using the JAVA-interfaces, you must ensure the right character set of the data.

If you have an ASCII platform with JDK, then you can also use XML components and operate on the BS2000 Oracle database using a JDBC connection.

> **See Also:**
>
> Java in the Database for more information about the encoding considerations

13
Oracle Net Services

This chapter describes Oracle Net Services and its implementation in the BS2000 and POSIX program environment. It supplements the *Oracle Database Net Services Administrator's Guide* with BS2000-specific information about the following topics:

- Oracle Net Protocol Support
- Oracle Network Security
- Shared Server Architecture
- Configuring the Network
- Troubleshooting Oracle Net Services

13.1 Oracle Net Protocol Support

Oracle Net Services supports network communication between a client application and a remote or a local database running on a variety of operating systems.

Oracle Net Services allows the database servers and the client applications, or servers acting as clients, to run on separate systems and provides a means for moving data between the nodes on a network. For example, a UNIX or Windows user can run applications that access and manipulate data in a remote Oracle database running on a BS2000 system.

Oracle Net Services is also used for inter-process communication, if a client and the requested database service are running on the same machine.

As mentioned in the Concepts and Architecture chapter, Oracle programs can run in both the BS2000 and the POSIX program environment. Depending on the program environment, the Oracle protocol adapter uses different socket implementations, such as, the BS2000 sockets and the POSIX sockets. This causes differences in the protocols used for the local inter-process communication.

Oracle Net Services supports the following network protocols:

- TCP/IP (version 4 and version 6)
- TCP/IP with SSL
- ISO Domain (in the BS2000 program environment only)
- Unix Domain (in the POSIX program environment and optionally in the BS2000 program environment)

Oracle Net Services automatically detects the program environment and uses the appropriate sockets implementation. You can also specify Oracle Net Services to use the POSIX sockets instead of the BS2000 sockets. Use the variable `NT_IPC_PROTOCOL_UNIX` to switch to the POSIX sockets with the Unix Domain protocol for local communication:

```
NT_IPC_PROTOCOL_UNIX={TRUE|FALSE} Default: FALSE
```

The POSIX sockets provide a smarter handling of the hand-off mechanism, which is required for the support of the Database Resident Connection Pooling (DRCP) feature. To ensure the use of the correct protocol for inter-process communication in both program environments, you must set the variable to the same value in both the POSIX and the BS2000 program environment.

This section contains the following topics:

- About Bequeath Protocol
- About IPC Protocol Support
- About TCP/IP Protocol Support
- About TCP/IP with SSL Protocol

> **Note:**
>
> The InterProcess Communication (IPC) protocol requires the same socket implementation for both the client and the server.

13.1.1 About Bequeath Protocol

The Bequeath technique enables clients to connect to a database without using the network listener. Oracle's Bequeath protocol internally spawns a server process for each client application. It does the same operation that a remote network listener does for the connection locally.

The Bequeath (BEQ) protocol is available in both program environments so that a client running in the POSIX shell can connect via the BEQ protocol to a server running in the BS2000 program environment. Ensure that the environment variable `NT_IPC_PROTOCOL_UNIX` has the same value in both program environments.

You can also use the POSIX sockets in the BS2000 program environment. In this case Oracle Net Services uses the protocol `PF_UNIX` for local connections and for handing off a connection.

By default, Oracle Net Services uses the BS2000 sockets in the BS2000 program environment, which is, `NT_IPC_PROTOCOL_UNIX=FALSE`. If you configure Oracle Net Services to use POSIX sockets in the BS2000 program environment, that is if `NT_IPC_PROTOCOL_UNIX=TRUE`, then ensure that the environment variable `NT_IPC_PROTOCOL_UNIX` is set to the same value for both the client and the local server.

13.1.1.1 Overview of the Bequeath Protocol

The Bequeath protocol:

- Does not use a network listener. Therefore, listener configuration is not required.
- Automatically spawns a dedicated server.
- Is used for local connections where an Oracle Database client application, such as SQL*Plus, communicates with an Oracle Database instance running on the same computer.

- Only works in the Dedicated Server mode. It cannot be used in the Shared Server mode.

> **Note:**
>
> If clients are running in a user ID that is different from the `DBA` user ID, then Oracle recommends using a net service name to connect through a listener to the destination database.

13.1.2 About IPC Protocol Support

This section introduces Oracle's Inter Process Communication (IPC) protocol support for inter-process calls. It is used to map the functionality of IPC to Oracle's Protocol Support Layer. Oracle Utilities and products running in the POSIX shell cannot use the IPC protocol if the variable `NT_IPC_PROTOCOL_UNIX` is set to `FALSE`.

It includes the following topics:

- Overview of IPC
- Using the IPC Protocol

13.1.2.1 Overview of IPC

On BS2000 systems, the IPC protocol is used for local inter-process communication. The Oracle Protocol Support for IPC uses the ISO functionality of the BS2000 sockets if `NT_IPC_PROTOCOL_UNIX=FALSE`. If POSIX sockets are also used in the BS2000 program environment, that is, if `NT_IPC_PROTOCOL_UNIX=TRUE`, then the Oracle Protocol Support for IPC uses the UNIX protocol family.

The client process initiates its IPC connection with the remote process by specifying a `KEY` that describes the listening process. Once the connection is established, the two communicating processes send and receive data through a continuous byte stream.

13.1.2.2 Using the IPC Protocol

The IPC protocol enables applications to integrate with the Inter Process Communication method on a local host. The following is the syntax for using IPC protocol:

```
(DESCRIPTION=
   (ADDRESS=
                (PROTOCOL=IPC)
               (KEY=alphanumeric)
            )
)
```

where:

`PROTOCOL` specifies the supported protocol. For IPC, the value is `IPC`.

`KEY` specifies the listen endpoint. A string of at most 32 characters: [a...z], [A...Z], [0...9], '.', '-', '_', '$'

The following is an example of an IPC ADDRESS that specifies a server on a local host:

```
(DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=IPC)
       (KEY=ORCL)
   )
)
```

> **Note:**
>
> If the IPC protocol is specified by a utility or user application running in the POSIX shell and if `NT_IPC_PROTOCOL_UNIX=FALSE`, then Oracle Net Services fails to use the IPC protocol. The following error message is displayed:
>
> ```
> TNS-12557: TNS:protocol adapter not loadable
> ```

13.1.3 About TCP/IP Protocol Support

This section introduces Oracle's TCP/IP (Transmission Control Protocol/Internet Protocol) protocol support, which is used to map the functionality within TCP/IP to Oracle's Protocol Support Layer.

The specific TCP/IP connection parameters are part of the `ADDRESS` keyword-value pair. The three TCP/IP–specific parameters can be entered in any order within the `ADDRESS` construct. The syntax used by Oracle's TCP/IP protocol support is:

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

where:

`PROTOCOL` specifies the supported protocol. For TCP/IP, the value is `TCP`.

`HOST` specifies the host name or the host's IP address.

`PORT` specifies the TCP/IP port number.

The following is an example of the TCP/IP ADDRESS specifying a client on the `sales-server` host:

```
(DESCRIPTION=
 (ADDRESS=
     (PROTOCOL=TCP)
     (HOST=sales-server)
     (PORT=1521)
```

```
 )
)
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

13.1.4 About TCP/IP with SSL Protocol

The TCP/IP with Secure Socket Layer (SSL) protocol is supported in this release. SSL provides authentication, encryption, and data integrity using public key infrastructure (PKI). SSL stores authentication data, such as certificates and private keys, in an Oracle Wallet.

The connection parameters for TCP/IP with SSL protocol are the same as TCP/IP except for the protocol, which is TCPS:

```
(DESCRIPTION=
 (ADDRESS=
     (PROTOCOL=TCPS)
     (HOST=hostname)
     (PORT=port#)
 )
)
```

where:

- `PROTOCOL` specifies the supported protocol. For TCP/IP with SSL protocol, the value is TCPS.
- `HOST` specifies the host name or the host's IP address.
- `PORT` specifies the TCP/IP with SSL protocol port number.

The following is an example of the TCP/IP with SSL protocol ADDRESS specifying a client on the sales-server host:

```
(DESCRIPTION=
 (ADDRESS=
     (PROTOCOL=TCPS)
     (HOST=sales-server)
     (PORT=2484)
 )
)
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

13.2 Oracle Network Security

This section describes how to configure the data integrity and the cryptographic services of Oracle Network Security for Fujitsu BS2000.

For using either the data integrity, or the cryptographic services, or both, you must specify the appropriate parameters in the SQLNET.ORA file.

Use the following parameters to specify whether a service (example: crypto-checksumming or encryption) should be active:

```
SQLNET.CRYPTO_CHECKSUM_CLIENT
SQLNET.CRYPTO_CHECKSUM_SERVER
SQLNET.ENCRYPTION_CLIENT
SQLNET.ENCRYPTION_SERVER
```

Each of the preceding parameters defaults to REJECTED.

Each of the preceding parameters can have one of the following values:

Value	Meaning
ACCEPTED	Enables the security service if the other side of the connection specifies REQUESTED or REQUIRED and there is a compatible algorithm available on the other side. It is inactive otherwise.
REJECTED	Disables the security service, and the connection fails if the other side specifies REQUIRED.
REQUESTED	Enables the security service if the other side specifies ACCEPTED, REQUESTED, or REQUIRED and there is a compatible algorithm available on the other side. It is inactive otherwise.
REQUIRED	Enables the security service, and the connection fails if the other side specifies REJECTED or if there is no compatible algorithm on the other side.

Use the following parameters to control the algorithms that are made available for each service on each end of a connection:

```
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER
SQLNET.ENCRYPTION_TYPES_CLIENT
SQLNET.ENCRYPTION_TYPES_SERVER
```

The value of each of these parameters can be either a list of algorithm names in parenthesis separated by commas or a single algorithm name.

Type	Values
Crypto checksum types	MD5, SHA1,SHA256, SHA384, SHA512
Encryption types	3DES112, 3DES168, AES128, AES192, AES256, DES, DES40, RC4_40, RC4_56, RC4_128, RC4_256

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Net Services Reference*
- *Oracle Database Security Guide*

13.3 Shared Server Architecture

The initialization parameters that control the shared server architecture are as follows:

- LOCAL_LISTENER
- DISPATCHERS
- MAX_DISPATCHERS
- SHARED_SERVERS
- MAX_SHARED_SERVERS
- SHARED_SERVER_SESSIONS
- CIRCUITS

The shared server architecture and the dedicated server architecture can work concurrently in an instance. Provide information in the connect descriptor to indicate whether a connecting application should use the shared server or the dedicated server architecture. By default, the listener process uses the shared server architecture. If you want the application to use the dedicated server architecture, then you must set USE_DEDICATED_SERVER=ON in the SQLNET.ORA file or specify a net service name with the parameter SERVER in the used naming method. The SQLNET.ORA parameter USE_DEDICATED_SERVER=ON overwrites the SERVER parameter.

The following example shows how to reference a dedicated server in a shared server configuration by using a specially defined net service name:

```
FINANCE_DED=(DESCRIPTION=
                     (ADDRESS=
                          (PROTOCOL=TCP)
                          (HOST=sales-server)
                          (PORT=1521))
                     (CONNECT_DATA=
                          (SERVICE_NAME=sales.us.example.com)
                          (SERVER=dedicated)))
```

To choose between using the shared server and the dedicated server architecture, you must consider the CPU overhead versus resource allocation, such as, tasks, memory, and so on. Use the shared server architecture if many clients work only occasionally with an Oracle database. Use the dedicated server architecture if only a few clients work intensively with an Oracle database. Before you decide, consider using the information in the following shared server dynamic tables:

- V$DISPATCHER
- V$QUEUE
- V$SHARED_SERVERS
- V$SHARED_SERVER_MONITOR

See Also:

- *Oracle Database Net Services Administrator's Guide* and *Oracle Database Administrator's Guide* for detailed information about the shared server architecture
- *Oracle Database Net Services Reference* for more information on how to reference a dedicated server in a shared server configuration
- *Oracle Database Administrator's Guide* for more information about shared server dynamic tables

13.4 Configuring the Network

You can either use the Easy Connect Naming Method to connect to the database or the Local Naming Method. When you use the Local Naming Method, Oracle recommends to configure clients for the use of service names that are easy to remember aliases for database addresses and match the address preconfigured in each system's `LISTENER.ORA` file. The client uses these addresses to connect to the network listener, which routes the connection request to the required service. During a connection, a client passes the service name to which it wants to connect.

The `LISTENER.ORA` file identifies and controls the behavior of the network listener that listens for services on the system. This file includes:

- Network listener descriptors and addresses
- Services the listener is listening for
- Various control parameters

Client configuration is accomplished by creating a list of net service names with addresses of network destinations through the local naming parameter file `TNSNAMES.ORA` or an LDAP compliant directory server.

13.4.1 About Using Easy Connect Naming Method

The Easy Connect naming method, can be used to connect to a database without the need to configure service names in the `TNSNAMES.ORA` configuration file. For using the Easy Connect naming method, ensure that `EZCONNECT` is listed in the client's configuration file parameter for naming adaptors `NAMES.DIRECTORY_PATH`.

Easy Connect naming is suitable for small and simple environments.

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for more information about the Easy Connect naming method

13.4.2 About Using the Local Naming Method

Local naming refers to the method of resolving a service name to a network address by using information configured on each individual client in a `TNSNAMES.ORA` configuration file. For using the local naming, ensure that `TNSNAMES` is listed in the client's configuration file parameter for naming adaptors `NAMES.DIRECTORY_PATH`.

Local naming is most appropriate for simple distributed networks with a small number of services that change infrequently.

13.4.3 About Using the Directory Naming Method

Directory Naming refers to the method of resolving a service name to a network address by using a directory server. For using a directory server, ensure that `LDAP` is listed in the client's configuration file parameter for naming adaptors `NAMES.DIRECTORY_PATH` and that the target address of the directory server is configured in the `LDAP.ORA` parameter file. For example:

```
# LDAP.ORA Network Configuration File: network.admin.ldap.ora
DEFAULT_ADMIN_CONTEXT = ""
DIRECTORY_SERVERS= (ldap_server:389:636)
DIRECTORY_SERVER_TYPE = Your Internet Directory Type
```

Where `DIRECTORY_SERVER_TYPE` specifies the type of the directory server that is being used. It can have the following values:

- `oid` for Oracle Internet Directory
- `ad` for Microsoft Active Directory

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* and *Oracle Database Net Services Reference* for more information

13.4.4 Customizing Oracle Net Listener Configuration

Before starting the listener, you must set up the configuration file `LISTENER.ORA`. This file includes the address of the listener and various control parameters used by the listener. When a listener accepts a connection request of a client, it routes the connection to a database server. In case of a dedicated server, the listener starts the server using the bequeath protocol and inherits the connection. Therefore, the listener should run in the DBA user ID of the Oracle database.

Usually the listener runs in the same DBA user ID as the Oracle database instance. If you have Oracle database instances running in different DBA user IDs, then you can configure a single listener to support all database services. In this case the listener

must know the LOGON authorization parameters, USER-ID, ACCOUNT, and PASSWORD of the foreign DBA user ID where Oracle database server tasks must be started. Use the Oracle environment variables *sid*_USER, *user*_ACCOUNT, and *user*_PASSWORD as described in the following table to specify the required LOGON authorization parameters:

Parameter	Meaning
BGJPAR	Parameters for ENTER jobs
*sid*_BGJPAR	Parameters for ENTER jobs identified by *SID*
*sid*_USER	The user ID under which the job should run
*user*_ACCOUNT	Account of the target user ID
*user*_PASSWORD	Password of the target user ID

> **Note:**
>
> In order to avoid storing a password in the ORAENV file, you can use SECOS, which is Fujitsu's Security Control System for BS2000. SECOS supports the facility to grant the listener's user ID the right to start batch jobs in the DBA user ID. Thus the environment variable *sid*_USER is the only variable that must be set.

The following example of an ORAENV file configured for a central listener process shows how the parameters work. The listener can share this ORAENV file with an instance, which runs under the same user ID. For a better understanding, we assume that the listener and the instances DEM0 and DEM1 are running under the user ID ORACDEM1 while the instance DEM2 is running under the user ID ORACDEM2. Define the following parameters:

```
BGJPAR=J-C=JCBORA,START=IMME,CPU-LIMIT=NO,LOGGING=*NO
DEM1_BGJPAR=J-C=JCBDEM1,START=IMME,CPU-LIMIT=NO
DEM2_USER=ORACDEM2
ORACDEM2_ACCOUNT=01234
ORACDEM2_PASSWORD=ORACLE
```

The listener always runs the same sequence to look up the parameters *sid*_BGJPAR and *sid*_USER. If the value for *sid*_BGJPAR is not found, then the listener uses the value given by the parameter BGJPAR. If a user ID is given by *sid*_USER, then the listener tries to get the processing admission from the parameters user_ACCOUNT and user_PASSWORD. For the given ORAENV, you get the following scenarios for the listener:

- The listener should start a server for the instance DEM0. Because the parameters DEM0_BGJPAR and DEM0_USER are not defined the listener starts the server for the instance DEM0 under the user ID ORACDEM1 with the start parameters defined by BGJPAR.
- If a server for the instance DEM1 must be started, then the listener looks for the parameters DEM1_BGJPAR and DEM1_USER. In this case the parameter DEM1_BGJPAR can be evaluated, whereas, the evaluation of DEM1_USER failed because this parameter is not defined. Therefore, the listener adds the start parameters "J-C=JCBDEM1,START=IMME,CPU-LIMIT=NO" to the ENTER-PROCEDURE command and starts the job under the user ID ORACDEM1.

- Now a server for instance DEM2 must be started. The listener looks for the parameters DEM2_BGJPAR and DEM2_USER. The parameter DEM2_BGJPAR is not defined so that the listener uses the start parameters defined by BGJPAR. On the other hand the parameter DEM2_USER can be evaluated successfully and returns the value ORACDEM2. Now the listener tries to get the processing admission by evaluating the parameters ORACDEM2_ACCOUNT and ORACDEM2_PASSWORD. The listener starts the server job under the user ID ORACDEM2 with the ENTER-PROCEDURE parameters "J-C=JCBORA,START=IMME,CPU-LIMIT=NO,LOGGING=*NO".

Start the listener using the Listener Control Utility LSNRCTL:

```
/CALL-PROCEDURE sid.P.ORAENV
/START-EXECUTABLE (*LINK(ORALOAD),LSNRCTL)
```

When the enter options prompt is displayed, press **ENTER** to get to the LSNRCTL prompt. Enter the following command to start the listener:

```
LSNRCTL> START listener-name
```

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

13.4.5 Configuration of the Client

Configuration of network clients involves adding or editing parameters in the client configuration file SQLNET.ORA and, depending on the used naming method, in the configuration file LDAP.ORA or TNSNAMES.ORA.

> **See Also:**
>
> *Oracle Database Net Services Reference* for more information about the configuration parameters

13.4.6 Testing the Configuration of the Client

After you have verified the network connections, verify the connections to the desired Oracle Database systems using the TNSPING utility:

```
/CALL-PROCEDURE sid.P.ORAENV
/START-EXECUTABLE (*LINK(ORALOAD),TNSPING)
```

When the enter options prompt displays, enter the net service name for the database service that you have specified in the naming service. If everything works fine, then a message similar to the following is returned:

```
TNS Ping Utility for BS2000: Version 19.0.0.0 -
Production on 11-MAR-2020 10:28:12
Used parameter files: network.admin.sqlnet.ora
Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = sales-server)(PORT = 3055)))
```

```
(CONNECT_DATA = (SERVICE_NAME = sales.us.example.com)))
OK (30 msec)
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

13.5 Troubleshooting Oracle Net Services

The following is a list of error messages and steps to fix the errors:

1. Listener could not be started. `LSNRCTL` returns the following error message:

   ```
   LSNRCTL> start
   Starting /BS2/$ORACINST.tnslsnr: please wait...

   TNS-12547: TNS:lost contact
   TNS-12560: TNS:protocol adapter error
   TNS-00517: Lost contact
   BS2000 Error: 145: Connection timed out
   LSNRCTL>
   ```

   - Ensure that the subsystem `POSIX` is up and running.
   - Ensure that the `BCAM` Light Weight Resolver `LWRESD` is properly configured and running.
2. Listener could not open the log file.
   - Check the `listener.log` file in the ADR subdirectory for `tnslsnr`. For example:

     `/u01/app/oracle/diag/tnslsnr/hostname/listener/trace/listener.log`
   - If ADR is disabled, then check if the `listener.log` file in the BS2000 file system, for example, `NETWORK.LOG.LISTENER.LOG`, is accessible and readable.
   - Verify the listener log file in the BS2000 file system using the BS2000 SDF command `REPAIR-DISK-FILES`.
   - If you are not able to fix the listener log file, then delete the file.
3. A client reports ORA-12545: Connect failed because target host or object does not exist
   - Check the naming service, if the host name is well known in the TCP/IP network.
   - Use a Fully Qualified Domain Name (`hostname.domain`)
   - Ensure that the `BCAM` Light Weight Resolver `LWRESD` is properly configured and running.
4. A client reports ORA-12535: TNS operation timed out
   - If you use the IPC protocol, then check the Connection Timeout parameter of `BCAM` (use the `BCSHOW` command). This parameter should be set to at least 600 seconds.
5. A client reports ORA-03113: end-of-file on communication channel
   - Check if the `SQLNET.EXPIRE_TIME` parameter is set for the server. If the parameter is set, then check the `BCAM LETTER-TIMER` using the `BCSHOW` command. If the `LETTER-TIME` is less than the `SQLNET.EXPIRE_TIME`, then the

data that is sent by the server to see if the client is running may not be read during their lifetime, which is limited by the `LETTER-TIME`. As a result, the client logs a broken pipe in the `SQLNET.LOG` file:

```
ns main err code: 12547
ns (2)  err code: 12560
nt main err code: 517
nt (2)  err code: 32
nt OS   err code: 0x0040002c
```

You can solve this problem by setting the `LETTER-TIMER` to infinite.

Part IV
Application Development

This part summarizes topics for application developers:

- Database Applications
- External Procedures
- Globalization Support

14
Database Applications

This chapter contains the following topics:

14.1 Overview of Database Applications

Oracle Programmatic Interfaces are tools for application designers who want to use SQL statements to access an Oracle database from within high-level language programs. The following types of programmatic interfaces are available:

- The Precompiler Interface, which is a programming tool that enables you to embed SQL statements in high-level language source code.
- The Oracle Call Interface (OCI), that enables to create high-level language applications that use function calls to access an Oracle database and control all phases of SQL statement execution.

On BS2000, the Oracle Database precompilers support programs written in C, C++, and COBOL programming languages.

This section contains the following topics:

> **See Also:**
>
> - *Oracle Database Programmer's Guide to the Oracle Precompilers* for detailed information about Oracle Precompilers
> - *Pro*C/C++ Programmer's Guide* or *Pro*COBOL Programmer's Guide*

14.1.1 Architecture of the Programmatic Interfaces

All precompiler and Oracle Call Interface (OCI) applications are linked with a small stub module. This stub module dynamically loads the SQL runtime system of the

Oracle Database precompilers from the ORALOAD library. Programs written in the following languages can be combined:

- Pro*C/C++
- Pro*COBOL (COBOL85 and COBOL2000)

> **Note:**
>
> OCI C and OCI COBOL programs cannot be combined; any attempt to do so results in execution errors. The entries into Oracle Database used by OCI C and OCI COBOL (for example, OLOGON) have identical names but different argument lists. For OCI COBOL, all arguments are by reference, that is, the parameter list contains all pointers. For OCI C, the numeric arguments are by value.

Oracle Database precompilers generate different SQLLIB function names for different languages. The following names are used:

- SQ0XXX: COBOL
- sq2xxx: C

14.1.2 PL/SQL Support

The precompilers support PL/SQL. When using PL/SQL, you must specify SQLCHECK=FULL or SQLCHECK=SEMANTICS on the precompiler option line. The default is SQLCHECK=NONE. When requesting SQLCHECK, the precompiler must connect to a database. So, ensure that you provide the necessary connection information. (You may also want to set the DEFAULT_CONNECTION variable in the ORAENV file).

When SQLCHECK=SEMANTICS or SQLCHECK=FULL is specified, you must also specify USERID=*username/password*.

> **See Also:**
>
> *Oracle Database PL/SQL Language Reference*

14.1.3 Building and Running a Programmatic Interface Application

To build and run a programmatic interface application, perform the following steps:

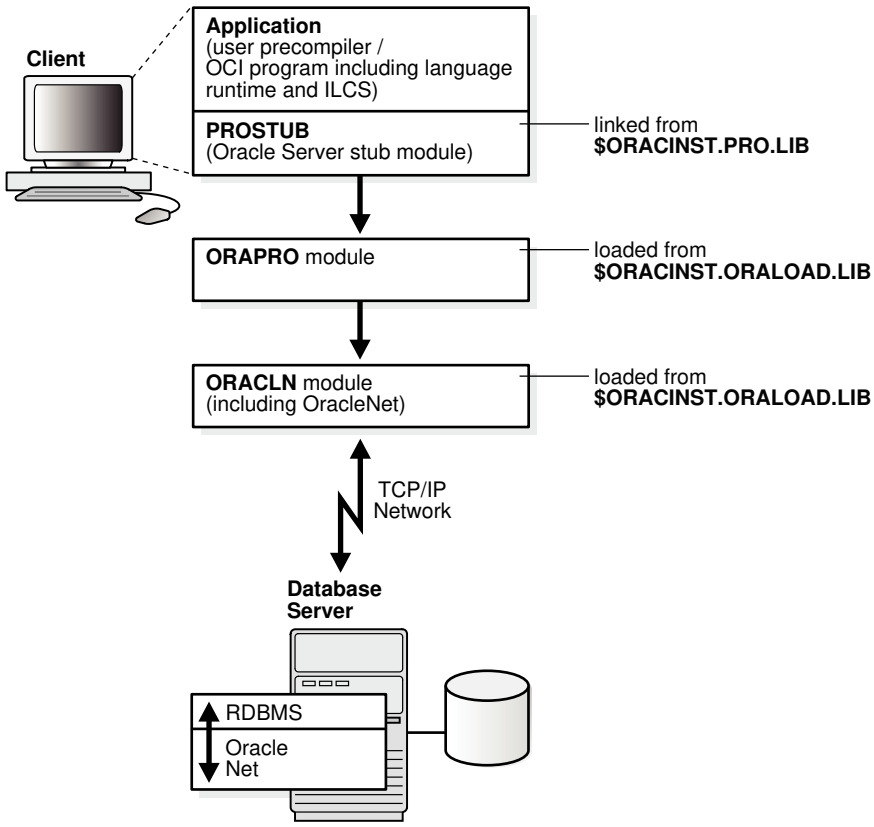
1. Edit your source code, including embedded SQL, as outlined in the generic precompiler documentation.
2. Precompile the source with the corresponding precompiler.

> **Note:**
>
> You must use WE8BS2000 as client character set during precompilation (set in `ORAENV` file). Any other character set might lead to problems with the concatenation sign ("||").
>
> You do not need to precompile if you build an OCI C or an OCI COBOL application.

3. Compile the application.
4. Link the application, including the stub module `PROSTUB` from the `$ORACINST.PRO.LIB` library.
5. Call the environment definition file. Refer to "Calling the Environment-Definition File" for details.
6. Run the application with the `START-EXECUTABLE` command. The supporting Oracle Database module is dynamically loaded from the `ORALOAD` library.
7. You can find sample BS2000 procedures for precompiling, compiling, and linking in the installation user ID: `$ORACINST.P.PROC`, `$ORACINST.P.PROCOB`, and `$ORACINST.P.PROLNK`.

Figure 14-1 illustrates the sequence of events outlined in the preceding numbered list and how the programmatic interfaces use the program libraries.

Figure 14-1 Usage of Program Libraries by Programmatic Interfaces



For more information, see the specific notes for the programmatic interfaces in this chapter.

14.2 Precompiler Applications

Learn about precompiler applications in this section. It includes the following topics:

- About Using Precompilers
- Precompiler Pro*C/C++
- Precompiler Pro*COBOL

14.2.1 About Using Precompilers

Oracle Database precompilers on BS2000 support LMS libraries for the files mentioned in this section. This section includes the following topics:

- Include Files
- User-Specific Configuration Files
- Input, Output, and List-files
- Additional Remarks About Using Precompilers

This functionality helps saving disk resources and provides clarity by grouping files in different libraries.

All LMS library elements that you use must be of element type "S". The precompilers generate elements of type "S" if libraries are used. When you use LMS library elements, the precompilers build temporary files with the prefix "#T.", which are deleted when the preprocessing completes successfully.

When you use LMS library elements, the element name that you specify must be the full element name including the suffix. The precompilers do not append the suffix to the element name.

14.2.1.1 Include Files

All standard include files are shipped in the LMS library, `$ORACINST.PRO.INCLUDE.LIB`. You must specify this LMS library or a user-defined include library for the `EXEC SQL INCLUDE` statements. Use the `INCLUDE` precompiler option, as follows:

```
* INCLUDE=$ORACINST.PRO.INCLUDE.LIB \
* INCLUDE=mylibrary
```

where *`mylibrary`* is the BS2000 file name of the user-defined library, such as `PROC.INCLIB`.

> **Note:**
>
> The order in which you specify the different `INCLUDE` options affects the performance of precompilation. You should place the commonly used files before the rarely used ones.

14.2.1.2 User-Specific Configuration Files

You can also specify a user-specific configuration file as an LMS-element using the following syntax:

```
* CONFIG=my_config_lib[config_element]
```

where *my_config_lib* is the BS2000 file name of the configuration library and *config_element* is the full name of the element.

You must use brackets when specifying the configuration element, as shown in the following example.

```
* CONFIG=CONFIG.LIB[PROCOB.CFG]
```

14.2.1.3 Input, Output, and List-files

Besides using BS2000 files, you can also use the LMS library elements for precompiler I/O using the `INAME`, `ONAME`, and `LNAME` options.

If you do not specify a library file name and an element, then the Oracle Database precompilers generate BS2000 ISAM files by default. The only option that you must enter is the `INAME` option. This can be either a BS2000 file name, or a library file name and the name of an element from the LMS library.

For example:

```
* INAME=my_input_lib[my_element] \
* ONAME=my_output_lib[my_element] \
* LNAME=my_list_lib[my_element] \
```

where *my_input_lib* is the BS2000 file name of the particular library and *my_element* is the name of the element including the specific suffix.

> **Note:**
>
> You must use brackets when specifying the LMS library element.

In the following example, Pro*C generates an output file with the name `SAMPLE.C` as the `ONAME` option has been omitted:

```
* INAME=INPUT.LIB[SAMPLE.PC] \
* LNAME=LIST.LIB[SAMPLE.LST]
```

14.2.1.4 Additional Remarks About Using Precompilers

The following are some additional remarks on this release of Oracle Database 19c for Fujitsu BS2000:

- Only compilers and compiler versions supporting the ILCS Standard Linkage are supported. If the Oracle Database detects a call from a user program not using the Standard Linkage conventions, then it terminates the task and displays the message number 5002 or 5003.

- If ONAME is not specified when starting a precompiler, then the precompiler generates a default name, which consists of the last part of INAME with the relevant suffix. For example, if the name of the C program you want to compile is *MYPROG.PERS.TEST.PC*, and if ONAME is omitted, then Pro*C generates an output file with the name TEST.C.
- If you work with float variables, then you may encounter rounding problems. The workaround is to declare the float variables as double variables instead.

14.2.2 Precompiler Pro*C/C++

This section describes the procedure for using Pro*C/C++. It includes the following topics:

- Starting Pro*C
- Pro*C Include, System Configuration and Demo Files
- SQLLIB Calls
- Linking Pro*C Programs
- The Pro*C SQLCPR.H Header File
- UTM Applications

14.2.2.1 Starting Pro*C

To start the Pro*C precompiler, enter the following:

```
/START-EXECUTABLE (*LINK(ORALOAD),PROC)
* INAME=myprog.PC ONAME=myprog.C [options]
```

where:

myprog is the name of the C program.

options specifies Pro*C options. For a list and description of the valid options, see *Pro*C/C++ Programmer's Guide*.

> **Note:**
>
> You must use a separate precompiler option INCLUDE for each path you want to specify, unlike as described in *Pro*C/C++ Programmer's Guide*. A list as allowed for the option SYS_INCLUDE may cause the precompiler to loop.

14.2.2.2 Pro*C Include, System Configuration and Demo Files

The Pro*C include files, demo files, and system configuration files are shipped under:

```
$ORACINST.PRO.INCLUDE.LIB
$ORACINST.C.DEMO.*.PC
$ORACINST.UTM.DEMO.*.PC
$ORACINST.CONFIG.PCSCFG.CFG
```

An example of a BS2000 procedure for precompilation and compilation is included in the Oracle Database software under the name $ORACINST.P.PROC.

14.2.2.3 SQLLIB Calls

If you want to program explicit C calls to `SQLLIB` functions, then you must call `sq2`*`xxx`* instead of `sql`*`xxx`*. For example, call `sq2cex` instead of `sqlcex`.

14.2.2.4 Linking Pro*C Programs

To link a Pro*C program, you need:

- The common runtime environment, `CRTE`.
- The Pro* library `$ORACINST.PRO.LIB`, which contains the stub module, `PROSTUB`.

To link your program, you must create your user-specific link procedure. An example of such a link procedure is included in the Oracle Database software with the name, `$ORACINST.P.PROLNK`.

14.2.2.5 The Pro*C SQLCPR.H Header File

If you are making calls to Pro*C functions, such as `sq2cls()` or `sq2glm()`, then you can include the `SQLCPR.H` file into the C programs to verify that the functions calls are correct.

In the Pro*C programs, add the following line:

```
EXEC SQL INCLUDE SQLCPR
```

as you would for `SQLCA` or `SQLDA`.

14.2.2.6 UTM Applications

You can use Pro*C to write UTM program units.

> **See Also:**
>
> openUTM Database Applications

14.2.3 Precompiler Pro*COBOL

This section describes the procedure for using Pro*COBOL. It includes the following topics:

- Starting Pro*COBOL
- Pro*COBOL Include, System Configuration, and Demo Files
- SQLLIB Calls
- Linking Pro*COBOL Programs
- openUTM Applications
- Additional Information About Pro*COBOL Constructs

14.2.3.1 Starting Pro*COBOL

To start the Pro*COBOL precompiler, enter the following commands:

```
/START-EXECUTABLE (*LINK(ORALOAD),PROCOB)
* INAME=myprog.PCO ONAME=myprog.COB [options]
```

where:

myprog specifies the COBOL program.

options specifies the Pro*COBOL options.

> **Note:**
>
> The Pro*COBOL option MAXLITERAL defaults to 180, and not 256, as shown in the *Pro*COBOL Programmer's Guide*. The option FORMAT=TERMINAL is not supported.

> **See Also:**
>
> *Pro*COBOL Programmer's Guide* for PRO*COBOL options

14.2.3.2 Pro*COBOL Include, System Configuration, and Demo Files

The Pro*COBOL include files, demo files, and system configuration files are shipped under:

```
$ORACINST.PRO.INCLUDE.LIB
$ORACINST.COBOL.DEMO.*.PCO
$ORACINST.UTM.DEMO.*.PCO
$ORACINST.CONFIG.PCBCFG.CFG
```

An example of a BS2000 procedure for precompilation and compilation is included in your Oracle Database software under the name $ORACINST.P.PROCOB2000.

14.2.3.3 SQLLIB Calls

If you want to program explicit COBOL calls to SQLLIB functions, then call SQ0*XXX* instead of SQL*XXX*. For example, call SQ0ADR instead of SQLADR.

14.2.3.4 Linking Pro*COBOL Programs

To link a Pro*COBOL program, you need:

- The common runtime environment, CRTE.
- The Pro* Library $ORACINST.PRO.LIB, which contains the stub module, PROSTUB.
- Unicode, which is only supported with COBOL2000. This might generate calls to the BS2000-Macro NLSCNV. To resolve the GNLCNV entry, use the BS2000 XHCS library.

> **See Also:**
>
> Fujitsu BS2000 manual *XHCS* for more information about the `GNLCNV` entry

To link your program, you must create your own user-specific link procedure. An example of such a link procedure is included in your Oracle Database software with the name, `$ORACINST.P.PROLNK`.

14.2.3.5 openUTM Applications

You can use Pro*COBOL to write openUTM (Universal Transaction Monitor) program units. Pro*C and Pro*COBOL program units can be combined in an openUTM application.

> **See Also:**
>
> openUTM Database Applications for more information

14.2.3.6 Additional Information About Pro*COBOL Constructs

When using Pro*COBOL, be careful about the following constructs with paragraphs and `EXEC` statements, because the precompiler generates a paragraph heading for the code generated from these `EXEC` statements:

Before precompiling	***After precompiling***
COB-LABEL1.	COB-LABEL1.
.	.
.	.
EXEC SQL....	SQL-LABEL1.
.	.
.	.
COB-LABEL2.	COB-LABEL2.

Before precompiling, the statement `PERFORM COB-LABEL1` runs the code in paragraph `COB-LABEL1` until the `COB-LABEL2` heading is reached. However, the precompiler generates a paragraph heading, `SQL-LABEL1`, for the code generated from the `EXEC SQL` statement.

As a result, after precompiling, `PERFORM COB-LABEL1` runs the code in the paragraph `COB-LABEL1` until the `SQL-LABEL1` heading is reached. The workaround for this problem is to use `SECTIONS` or to run `PERFORM COB-LABEL1 THRU COB-LABEL2`.

A `COPY` statement as the first statement in `WORKING-STORAGE SECTION` may result in a wrong code generation if copied structures must be continued by non-copied code. This is because the precompiler generates its data definitions before the first data definition of the source program. To avoid this, insert a `FILLER` definition as the first line in `WORKING-STORAGE SECTION`, as follows:

```
01 FILLER PIC X
```

14.3 Oracle Call Interface Applications

On Fujitsu BS2000, the Oracle Call Interface (OCI) supports the programming languages, C and COBOL.

When you use the set of host language calls that comprise the Oracle Call Interface, you can access the data in an Oracle database by programs written in the C and COBOL programming languages.

> **Note:**
>
> The precompiler products from Oracle offer a higher level interface to the Oracle Database. A single precompiler call is translated to several OCI calls. As the precompilers are easy to use, and in a few cases offer more or different functionality than OCI, you may prefer to use the precompilers for some applications.

> **See Also:**
>
> *Oracle Call Interface Programmer's Guide* for more information about OCI calls

14.3.1 Linking OCI Applications

To link OCI programs, you need:

- The common runtime environment, *CRTE*.
- The Pro* Library `$ORACINST.PRO.LIB`, which contains the stub modules `OCI$COB` and `PROSTUB`.

When linking OCI COBOL programs, `OCI$COB` must always be included **before** `PROSTUB`.

To link your program, you must create your own user-specific link procedure. An example of such a link procedure is included in your Oracle Database software with the name `$ORACINST.P.PROLNK`.

For example, to link your program, call the BS2000 procedure as follows:

`/CALL-PROCEDURE $ORACINST.P.PROLNK,`*`dir`*`,`*`module`*`,TYPE=OCIC`

or:

`/CALL-PROCEDURE $ORACINST.P.PROLNK,`*`dir`*`,`*`module`*`,TYPE=OCICOB`

where, the module to be linked is stored in *`dir`*`.LIB`.

The OCI include files and demo files are shipped under:

```
$ORACINST.RDBMS.DEMO.OCI.LIB
$ORACINST.RDBMS.DEMO.*.C
$ORACINST.RDBMS.DEMO.*.COB
```

14.4 The Object Type Translator

This section describes how to use the Object Type Translator (OTT) on BS2000. It includes the following topics:

- Starting Object Type Translator
- OTT System Configuration File

14.4.1 Starting Object Type Translator

The Object Type Translator (OTT) is based on Java and can only be started in the POSIX environment. You must use the JDBC thin driver to connect to the database. The connect string is specified in the `url` option, as follows:

```
url=jdbc:oracle:thin:@hostname:port:sid
```

In the following example, OTT will connect to the database with the service identifier `orcl`, on the host `myhost`, that has a TCP/IP listener on port 1521.

For example:

```
ott userid=username-for-scott/password-for-scott
url=jdbc:oracle:thin:@myhost:1521:orcl intype=demoin.typ
outtype=demoout.typ code=c hfile=demo.h
```

> **See Also:**
>
> *Pro*C/C++ Programmer's Guide* for more information about Object Type Translator

14.4.2 OTT System Configuration File

The OTT system configuration file is installed with the Oracle Database software, with the following name:

```
$ORACLE_HOME/precomp/admin/ottcfg.cfg
```

14.5 Oracle Database Applications in POSIX Program Environment

You can run application programs either in the BS2000 program environment or in the POSIX program environment. This section describes how you can build Oracle database applications that can run in the POSIX program environment.

You must precompile and compile the Pro* application or OCI application as described in the previous chapters.

When linking the application, you must include the stub module PROSTUBX from the $ORACINST.PROX.LIB library instead of PROSTUB and you must add the following lines in the BS2000 procedure for linking:

```
/SET-FILE-LINK BLSLIB01,$.SYSLNK.CRTE.POSIX
/SET-FILE-LINK BLSLIB02,$.SYSLIB.POSIX-SOCKETS.version_number
/SET-FILE-LINK BLSLIB03,$.SINLIB.POSIX-BC.version_number
```

> **Note:**
>
> The $.SYSLNK.CRTE.POSIX library must be the first library in the search order for the resolution of external references. Oracle recommends a search order as mentioned above.

To start an Oracle Database application in the POSIX environment by using BS2000 SDF commands, set the BS2000 SDF-P variable SYSPOSIX.PROGRAM-ENVIRONMENT to SHELL.

You can set additional POSIX environment variables by using the BS2000 SDF-P variable SYSPOSIX.

The following example shows how to set the SDF-P variable SYSPOSIX to run an application in the POSIX environment:

```
/DECL-VAR SYSPOSIX,TYPE=*STRUCT(DEF=*DYN),SCOPE=*TASK(STATE=*ANY)
/SET-VAR SYSPOSIX.PROGRAM-ENVIRONMENT='SHELL'
/SET-VAR SYSPOSIX.ORACLE-HOME='oracle_home_path'
/SET-VAR SYSPOSIX.ORACLE-SID='oracle_sid'
```

14.6 openUTM Database Applications

This section describes how to use the BS2000 transaction monitor openUTM for coordinated interoperation with Oracle Database 19c. The following topics are covered:

- Operation of Oracle Database Using openUTM Programs
- Distributed openUTM Files
- DBA Responsibilities
- Developing an Oracle Database/openUTM Application
- Troubleshooting

14.6.1 Operation of Oracle Database Using openUTM Programs

Universal Transaction Monitor (openUTM) controls the execution of user programs that can be used from a large number of terminals at the same time.

An openUTM application consists of a structured sequence of processing stages that are supplied with access rights for the specific user. These stages, in turn, consist of openUTM transactions that are carried out either in their entirety, or not at all.

If several users use openUTM at the same time, then simultaneous access to the shared database is also usually required. The database/data communications system (DB/DC system), Oracle Database/openUTM, synchronizes access by openUTM applications to Oracle Database, and ensures that the database and the transaction monitor are always in a shared consistent state. In the event of a system failure, the DB/DC system performs an automatic recovery, which ensures that the database remains in a consistent state that is synchronized with openUTM.

Synchronization of Oracle Database and openUTM is done through the XA interface. The XA interface is an X/Open interface for the coordination between database systems and transaction monitors.

> **See Also:**
>
> *Oracle Database Development Guide* for a description of the concepts of the XA interface

14.6.2 Distributed openUTM Files

When you install Oracle Database, as described in Oracle Database Installation and Deinstallation , openUTM related software and files are installed in the installation user ID. The distributed openUTM files include:

- `XAO.LIB`

  This file contains the connection module for the XA interface.

- The following files provide examples of procedures and programs:

```
UTM.DEMO.P.COMPILE.C
UTM.DEMO.P.COMPILE.COBOL
UTM.DEMO.P.KDCDEF
UTM.DEMO.P.KDCROOT
UTM.DEMO.P.PROBIND
UTM.DEMO.P.PROSTRT
UTM.DEMO.CSELEMP.PC
UTM.DEMO.SELDEP.PCO
UTM.DEMO.SELEMP.PCO
UTM.DEMO.UPDEMP.PCO
UTM.DEMO.ERRSQL.C
UTM.DEMO.ERRTXT.C
```

14.6.3 DBA Responsibilities

This section describes the responsibilities of the DBA or the administrator of the openUTM application.

The administrator of the openUTM application must define the open string for the XA interface with help from the application developer. This open string must be included in the openUTM start parameters.

The DBA must ensure that the data dictionary view `DBA_PENDING_TRANSACTIONS` exists. The DBA must also grant the `SELECT` privilege to the data dictionary view `DBA_PENDING_TRANSACTIONS` for all Oracle users specified in the XA open string. Use the following example to grant the `SELECT` privilege to user `scott`:

```
grant select on DBA_PENDING_TRANSACTIONS to scott;
```

The Oracle users are identified in the open string with the item `Acc`.

> **See Also:**
>
> "Defining an Open String" for more information

14.6.4 Developing an Oracle Database/openUTM Application

Oracle Database 19c on BS2000 supports openUTM V6.5 or higher. openUTM supports the XA interface. Oracle Database 19c on BS2000 coordinates with openUTM through this XA interface.

This section includes the following topics:

- How to Build an Oracle Database Application with openUTM
- Defining an Open String
- Using Precompilers with openUTM
- SQL Operations
- openUTM Operations

14.6.4.1 How to Build an Oracle Database Application with openUTM

The main steps involved in developing an Oracle Database application for coordinated inter-operation with openUTM are as follows:

- Building the openUTM program units
- Defining the configuration
- Translating the `KDCROOT` table module and openUTM program units
- Linking the openUTM application program
- Starting the openUTM application

Building the openUTM program units

Refer to the openUTM manual *Programming Applications with KDCS for COBOL, C, and C++*.

Defining the configuration

Refer to the following openUTM manuals:

- *Generating Applications*
- *Administering Applications*

An Oracle Database/openUTM application requires the following information for execution:

- Information about the application
- Username/password with access protection

- Information about the terminal and communication partners
- Information about the transaction codes (TAC's)

These properties collectively form the configuration, which is stored in the KDCFILE file. The configuration definition is carried out by the KDCDEF utility.

This section gives the descriptions for openUTM KDCDEF control statements that are important for connecting to the Oracle database. They are:

- DATABASE

  When the Oracle Database/openUTM application is generated, you must specify that openUTM communicates with the Oracle Database. Enter the following command to specify openUTM communication with the database:

  ```
  DATABASE TYPE=XA,ENTRY=XAOSWD
  ```

  where TYPE=XA specifies the use of the XA interface and ENTRY=XAOSWD specifies the name of the XA switch for the Oracle database (for dynamic registration).

- OPTION

  If you specify the corresponding GEN operand in the OPTION command, then the KDCDEF utility also produces the source-code for the KDCROOT table module.

- MAX

  Another important operand is APPLIMODE, which is specified in the MAX command. This determines the restart behavior after a system failure. The syntax of MAX is as follows:

  ```
  MAX APPLINAME=name[,APPLIMODE={S[ECURE]|F[AST]}]
  [,ASYNTASKS=number][...]
  ```

  APPLIMODE=SECURE means that openUTM continues after an application malfunction with a coordinated warm-start of the openUTM application and the Oracle database.

  If you specify APPLIMODE=FAST, then openUTM application restart is not executed, as openUTM does not store the restart information. In the event of an error, the application starts from scratch. Transactions that are still open after an openUTM-application malfunction are rolled back automatically.

See the UTM.DEMO.P.KDCDEF file for an example procedure for building the KDCFILE and the KDCROOT table module.

Translating the KDCROOT table module and openUTM program units

The source of the KDCROOT table module must be compiled with the BS2000 Assembler and the openUTM program units must be compiled with the corresponding programming language compilers. See the example procedure UTM.DEMO.P.KDCROOT for the compilation of the KDCROOT table module.

Linking the openUTM application program

The openUTM application program is produced by linking the KDCROOT table module with the openUTM program units.

You must include the stub module XAOSTUB:

```
INC-MOD LIB=$ORACINST.XAO.LIB,ELEM=XAOSTUB
```

> **Note:**
>
> Instead of writing the linking procedure, you should use the example procedure `UTM.DEMO.P.PROBIND` and apply modifications as needed.
>
> To write your own linking procedure, study the example carefully before writing.

Starting the openUTM application

An example procedure for starting the openUTM application is in the `UTM.DEMO.P.PROSTRT` file.

When starting the openUTM application, you must specify the start parameters for both openUTM and Oracle Database.

The openUTM start parameters are described in the openUTM manual *Using openUTM Applications on BS2000 Systems*.

The start parameter for using the XA interface for coordinated inter-operation with Oracle Database is:

```
.RMXA RM="Oracle_XA",OS="<ORACLE open string>"
```

14.6.4.2 Defining an Open String

This section describes how to construct an open string and includes the following topics:

- Required Fields
- Optional Fields
- Examples

The transaction monitor uses this string to connect to the database. The maximum number of characters in an open string is 256. Construct the string as follows:

```
Oracle_XA{+required_fields...}[+optional_fields...]
```

where the *required_fields* are:

- `Acc=P/user/access_info`
- `SesTm=session_time_limit`

and the *optional_fields* are:

- `DB=db_name`
- `MaxCur=maximum_no_of_open_cursors`
- `SqlNet=connect_string`
- `DbgFl=value_from_1_to_15`

> **Note:**
>
> - You can enter the required fields and optional fields in any order when constructing the open string.
> - All field names are case-insensitive, although their values may or may not be case-sensitive depending on the system.
> - You cannot use the "+" character as part of the actual information string.

14.6.4.2.1 Required Fields

The required fields for the open string are:

Item	Meaning
`Acc`	Specifies user access information.
`P`	Indicates that explicit user and password information is provided.
`user`	Specifies the Oracle Database user name.
`access_info`	Specifies the Oracle Database password.

For example, `Acc=P/`*`username-for-scott/password-for-scott`* indicates that the user and password information is provided.

Ensure that the user has the `SELECT` privilege on the `DBA_PENDING_TRANSACTIONS` table in the previous example.

> **Note:**
>
> For security reasons, openUTM supports the placeholders `*UTMUSER` and `*UTMPASS` for *`user`* and *`access_info`*. The values for these openUTM placeholders are specified through the openUTM `KDCDEF` generation. When the `xa_open` call is executed, openUTM replaces these placeholders with the generated values. Using these placeholders is mandatory in openUTM V6.4, but it is optional in openUTM V6.5.

Item	Meaning
`SesTm`	Specifies the maximum amount of time a transaction can be inactive before it is automatically deleted by the system.
`session_time_limit`	Specifies the maximum time limit in seconds between the start of a global transaction and the commit or roll back of this transaction.

14.6.4.2.2 Optional Fields

Optional fields for the open string are described in the following table:

Item	Meaning
`DB`	Specifies the database name.

Item	Meaning
db_name	Indicates the name used in Oracle Database precompilers to identify the database. Application programs that use only the default database for the Oracle Database precompiler, that is, application programs that do not use the AT clause in their SQL statements, must omit the DB=*db_name* clause in the open string. **Note:** This default database is specified in the ORAENV file by the environment variable ORASID. Applications that use explicitly–named databases should indicate that database name in their DB=*db_name* field. For example, DB=payroll indicates that the database name is payroll and that the application program uses that name in AT clauses.

For more information about precompilers, refer to the section "Using Precompilers with openUTM" later in this chapter.

Item	Meaning
MaxCur	Specifies the number of cursors to be allocated when the database is opened. It serves the same purpose as the precompiler option maxopencursors.
maximum_no_of_open_cursors	Specifies the number of open cursors.

For example, MaxCur=5 indicates that the process should try to keep five open cursors cached.

> **See Also:**
>
> *Oracle Database Programmer's Guide to the Oracle Precompilers* for more information about maxopencursors

Item	Meaning
SqlNet	Specifies the SQL*Net connect string.
connect_string	Specifies the string that is used to open a connection to the database. This can be any supported Oracle Net Services connect string.

For example:

SqlNet=MADRID_FINANCE indicates an entry in TNSNAMES.ORA. For more information, refer to Oracle Net Services in this guide.

Item	Meaning
DbgFl	Specifies if debugging should be enabled (debug flag). For more information refer to "About Debugging" section in this chapter.

14.6.4.2.3 Examples

This section contains examples of open strings for the XA interface.

> **Note:**
>
> If the string is longer than one line, then refer to the openUTM documentation for information about how to split the string.

For the bequeath protocol of Oracle Net Services:

```
Oracle_XA+Acc=P/username-for-scott/password-for-scott+SesTm=0+DbgFl=15
```

For other protocols of Oracle Net Services:

```
Oracle_XA+SqlNet=MADRID_FINANCE+Acc=P/username-for-scott/password-for-
scott+SesTm=0
Oracle_XA+DB=finance+SqlNet=MADRID_FINANCE+Acc=P/username-for-scott/password-for-
scott
+SesTM=0
```

The optional fields `LogDir`, `Loose_Coupling`, `SesWT`, and `Threads` are not supported.

> **See Also:**
>
> *Oracle Database Development Guide* for more information about the fields in the open string

14.6.4.3 Using Precompilers with openUTM

You can choose from the following options when interfacing with precompilers:

- Using Pro*C with the Default Database
- Using Pro*C with a Named Database

Run all the precompiler programs with the option `release_cursor` set to `no`. Precompiler programs may be written in C or COBOL. The precompiler Pro*C is used in the examples.

14.6.4.3.1 Using Pro*C with the Default Database

If Pro*C applications access the default database, then ensure that the `DB=db_name` field used in the open string is not present. The absence of this field indicates the default connection as defined in the `ORAENV` file, and only one default connection is allowed for each process.

The following is an example of an open string identifying a default Pro*C connection:

```
Oracle_XA+SqlNet=MADRID_FINANCE+Acc=P/username-for-scott/password-for-
scott+SesTm=0
```

Here, `DB=db_name` is absent, indicating an empty database identifier string.

The following is the syntax of a select statement:

```
EXEC SQL SELECT ENAME FROM EMP;
```

14.6.4.3.2 Using Pro*C with a Named Database

If Pro*C applications access a named database, then include the `DB=db_name` field in the open string. Any database you refer to must reference the same *db_name* specified in the corresponding open string.

An application may access the default database, as well as one or more named databases, as shown in the following examples.

For example, if you want to update an employee's salary in one database, the department number `deptno` in another, and the manager information in a third database, then you must configure the following open strings in the transaction manager:

`Oracle_XA+SqlNet=MADRID_FINANCE1+Acc=P/username-for-scott/password-for-scott+SesTm=0`

`Oracle_XA+DB=MANAGERS+SqlNet=MADRID_FINANCE2+Acc=P/username-for-scott/password-for-scott+SesTm=0`

`Oracle_XA+DB=PAYROLL+SqlNet=MADRID_FINANCE3+Acc=P/username-for-scott/password-for-scott+SesTm=0`

There is no `DB=db_name` field in the first open string.

In the application program, enter declarations such as:

```
EXEC SQL DECLARE PAYROLL DATABASE;
EXEC SQL DECLARE MANAGERS DATABASE;
```

Again, the default connection corresponding to the first open string that does not contain the *db_name* field, does not require a declaration.

When performing the update, enter statements similar to the following:

```
EXEC SQL AT PAYROLL update emp set sal=4500 where empno=7788;
EXEC SQL AT MANAGERS update emp set mgr=7566 where empno=7788;
EXEC SQL update emp set deptno=30 where empno=7788;
```

There is no AT clause in the last statement because it refers to the default database.

You can use a character host variable in the AT clause, as shown in the following example:

```
EXEC SQL BEGIN DECLARE SECTION;
db_name1 CHARACTER(10);
db_name2 CHARACTER(10)
EXEC SQL END DECLARE SECTION;
.
.
set db_name1 = 'PAYROLL'
set db_name2 = 'MANAGERS'
.
.
EXEC SQL AT :db_name1 UPDATE...
EXEC SQL AT :db_name2 UPDATE...
```

> **See Also:**
>
> *Pro*COBOL Programmer's Guide* and *Pro*C/C++ Programmer's Guide* that discusses concurrent logons

> **Note:**
>
> - openUTM applications must not create Oracle database connections of their own. Therefore, an openUTM user is not allowed to issue `CONNECT` statements within an openUTM program. Any work performed by them would be outside the global transaction, and may confuse the connection information given by openUTM.
> - SQL calls must not occur in the openUTM start exit routine, however may occur in the conversation exit routine “Vorgangs-Exit”.

14.6.4.4 SQL Operations

UTM application program units must use embedded SQL. Calls to the Oracle Call Interface (OCI) are not allowed.

The following SQL operations are discussed:

- CONNECT
- COMMIT
- ROLLBACK
- SAVEPOINT
- Cursor Operations
- Dynamic SQL
- PL/SQL
- Autocommit

14.6.4.4.1 CONNECT

A connection is implicitly established when the UTM task is started. This connection uses the data specified in the open string. Additional explicit `CONNECT` operations issued by the program units are not allowed.

14.6.4.4.2 COMMIT

An explicit `COMMIT` statement is not allowed in UTM program units. The openUTM automatically issues a `COMMIT` statement at `PEND RE`, `PEND FI`, `PEND SP`, or `PEND FC` operation.

14.6.4.4.3 ROLLBACK

An explicit `ROLLBACK` statement is not allowed in UTM program units. The openUTM automatically issues a `ROLLBACK` statement on encountering a `PEND ER`, `PEND RS`, `PEND FR`, or `RSET` operation.

14.6.4.4.4 SAVEPOINT

The `SAVEPOINT` statement is not allowed in UTM program units.

14.6.4.4.5 Cursor Operations

A cursor is valid only until a `PEND` is executed. Because of a possible task change during a `PEND KP`, `PEND PA`, or `PEND PR`, you cannot perform operations on a previously filled cursor such as `OPEN` or `FETCH` after a `PEND KP`, `PEND PA`, or `PEND PR`.

However, you can open and fetch a new cursor after `PEND KP`. The alternative to using `PEND KP` is to use the *PGWT*-call, which waits until it receives an input from the terminal, or to assign the same `TACCLASS` to subsequent programs after a `PEND PA` or `PR` operation.

> **See Also:**
>
> The openUTM manual, *Programming Applications with KDCS for COBOL, C and C++*

14.6.4.4.6 Dynamic SQL

You may use dynamic SQL as described in *Oracle Database Programmer's Guide to the Oracle Precompilers*.

14.6.4.4.7 PL/SQL

`COMMIT`, `ROLLBACK`, `CONNECT`, and `SAVEPOINT` statements are not allowed in PL/SQL programs running under UTM.

14.6.4.4.8 Autocommit

Avoid autocommit operations as they violate the synchronization between Oracle Database and UTM transactions. Take precautions when using the DDL operations, as these often contain implicit autocommits.

For example, DDL statements such as, `CREATE TABLE`, `DROP TABLE`, and `CREATE INDEX` are not allowed in a global transaction because they force the pending work to be committed.

14.6.4.5 openUTM Operations

This section describes the Oracle Database-specific points that you must consider when using UTM operations. It describes the effect of the `PEND` (Program Unit End), and `RSET` (Reset) operations of openUTM. These operations represent the common synchronization points between openUTM and the Oracle Database.

The following openUTM operations are discussed:

- RSET and PEND RS
- PEND ER and PEND FR
- PEND KP, PEND PR, and PEND PA
- PEND RE, PEND FI, PEND SP, and PEND FC

When you issue a `PEND` call, UTM calls the Oracle Database internally through the XA interface for synchronization. When the `PEND` takes place:

- The user dialog/transaction is detached from the executing task.
- Any resource that is still attached to the user is released.

14.6.4.5.1 RSET and PEND RS

Resetting a UTM transaction implies rolling back the Oracle Database transaction.

14.6.4.5.2 PEND ER and PEND FR

When using these calls to terminate a UTM transaction, the Oracle Database transaction is also rolled back.

14.6.4.5.3 PEND KP, PEND PR, and PEND PA

These operations only end a UTM dialog step without affecting the corresponding Oracle Database transaction.

14.6.4.5.4 PEND RE, PEND FI, PEND SP, and PEND FC

These `PEND` calls cause an implicit `COMMIT` to be executed. All cursors that are not explicitly closed, are closed.

14.6.5 Troubleshooting

This section discusses how to recover data if there are problems or a system failure. Both trace files and recovering pending transactions are discussed in the following sections:

- Trace Files
- About Debugging
- In-Doubt or Pending Transactions
- Oracle Database Tables of the SYS User

14.6.5.1 Trace Files

The Oracle XA library logs any error and tracing information to its trace file. This information is useful in supplementing the XA error codes. For example, it can indicate whether an open failure is caused by an incorrect open string, failure to find the Oracle Database instance, or a login authorization failure. The trace file is created in the BS2000 user ID, where the openUTM application runs. The name of the trace file is:

`ORAXALOG.`*`pid-db_name-date`*`.TRC`

where

pid is the process identifier (TSN).

db_name is the database name you specified in the open string field DB=*db_name*.

date is the date when the trace file is created.

14.6.5.1.1 Trace File Examples

Examples of trace files are discussed in this section.

The following example shows a trace file for an application's task with the TSN 1234 that was opened on April 2nd 1999. The DB field for this application was not specified in the open string when the resource manager was opened

```
ORAXALOG.1234-NULL-990402.TRC
```

The following example shows a trace file that was created on December 15th 1998 by the BS2000 task with the TSN 5678. The DB field was specified as FINANCE in the open string when the resource manager was opened.

```
ORAXALOG.5678-FINANCE-981215.TRC
```

Each entry in the trace file contains information that looks similar to:

```
1032.2: xa_switch rtn ORA-22
```

where 1032 is the time when the information is logged, 2 is the resource manager identifier, xa_switch is an internal identifier of the Oracle XA library, and ORA-22 is the returned Oracle database information.

14.6.5.2 About Debugging

You can specify the DbgFl (debug flag) in the open string.

Depending on the debugging level (low:DbgFl=1, high:DbgFl=15), you can get more or less debug entries in the trace file ORAXALOG.*pid-db_name-date*.TRC.

> **See Also:**
>
> *Oracle Database Development Guide* for more information

14.6.5.3 In-Doubt or Pending Transactions

In-doubt or pending transactions are transactions that have been prepared but not yet committed in the database. Generally, openUTM resolves any in-doubt or pending transaction. However, the Database Administrator may have to override an in-doubt transaction when working with UTM-F, that is, APPLIMODE=FAST. For example, when the in-doubt transaction is:

- Locking data that is required by other transactions.
- Not resolved in a reasonable amount of time.

> **Note:**
>
> Overriding in-doubt transactions can cause inconsistency between openUTM and the database. For example, if the DB transaction is committed by the Database Administrator and the openUTM application rolls back the transaction in the warm-start phase, then the Oracle Database cannot roll this committed transaction back, therefore, causing an inconsistency.

14.6.5.4 Oracle Database Tables of the SYS User

The following tables of the SYS user contain transactions generated by regular Oracle Database applications and Oracle Database/openUTM applications:

- DBA_2PC_PENDING
- DBA_2PC_NEIGHBORS
- DBA_PENDING_TRANSACTIONS
- V$GLOBAL_TRANSACTION

> **See Also:**
>
> For detailed information about how to use these tables, refer to the sections in the *Oracle Database Administrator's Guide* that discuss failures during two-phase commit and manually overriding in-doubt transactions

For transactions generated by Oracle Database/openUTM applications, the following column information applies specifically to the DBA_2PC_NEIGHBORS table:

- The DBID column is always xa_orcl.
- The DBUSER_OWNER column is always *db_namexa.oracle.com*.

Remember that the *db_name* is always specified as DB=*db_name* in the open string. If you do not specify this field in the open string, then the value of this column is NULLxa.oracle.com for transactions that are generated by Oracle Database/openUTM applications.

For example, use the following sample SQL statement to find out more information about in-doubt transactions that are generated by Oracle Database/openUTM applications.

```
SELECT * FROM DBA_2PC_PENDING p, DBA_2PC_NEIGHBORS n
WHERE p.LOCAL_TRAN_ID = n.LOCAL_TRAN_ID AND n.DBID = 'xa_orcl';
```

15
External Procedures

This chapter describes how to create an environment on BS2000, where external C procedure calls can operate. External JAVA methods are not supported on BS2000.

This chapter complements the chapter about External Procedures in the *Oracle Database Development Guide*.

> **Note:**
>
> The default configuration for external procedures does not require a network listener to work with Oracle Database and the `extproc` agent. The `extproc` agent is spawned directly by Oracle Database and eliminates the risks that the `extproc` agent might be spawned by Oracle Listener unexpectedly. This default configuration is recommended for maximum security.
>
> You can change the default configuration for external procedures and have the `extproc` agent spawned by Oracle Listener. To do this, you must perform additional network configuration steps.
>
> Having the `extproc` agent spawned by Oracle Listener is necessary if you use:
>
> The `AGENT` clause of the LIBRARY specification or the `AGENT IN` clause of the PROCEDURE specification such that you can redirect external procedures to a different `extproc` agent.

15.1 Loading External Procedures

This section complements the corresponding part in *Oracle Database Development Guide*. It shows how to use an external C procedure stored in a dynamic load library (DLL) with Oracle on BS2000. In this context a DLL is specified as a LMS library in the BS2000 environment or a shared object in the POSIX environment.

Perform the following steps to load external procedures:

- Define C Procedures
- Set Up the Environment
- Identify the DLL
- Publish External Procedures
- Run External Procedures

15.1.1 Define C Procedures

Define the C procedures using one of the prototypes.

Refer to *Oracle Database Development Guide* for the prototypes. Compile the C procedures either in the BS2000 environment or in the POSIX program environment.

BS2000 Program Environment

Compile the program using the BS2000 C/C++ compiler. The created LLM object must be stored in a LMS library. You must consider the default settings `LOWER-CASE-NAMES=*NO` and `SPECIAL-CHARACTERS=*CONVERT-TO-DOLLAR` of the C/C++ compiler option `MODULE-PROPERTIES`. These default settings cause the conversion of all lowercase letters in the entry names to uppercase and of all underscores (_) in the entry names to dollar signs ($).

POSIX Program Environment

Use the POSIX installation of the BS2000 C/C++ compiler to compile the program in the POSIX shell. You must consider that on default lower case letters in entry names will be translated to upper case letters and underscores in entry names will be translated to dollar signs. Use the option `-K llm_case_lower,llm_keep` to retain lower case letters and underscore characters when entry names are generated.

The example shows how to compile the source `C_concat.c`:

```
$ /usr/bin/cc -c -K llm_case_lower,llmcase_lower -B
extended_external_names -D_OSD_POSIX -I
/u01/app/oracle/product/19.3.0/dbhome_1/rdbms/public C_concat.c
```

Use the tool `genso` to generate a shared object. The following example illustrates how to create the shared object `C_utils.so` for the C procedure,

```
char *concat(ctx, str1, str1_i, str2, str2_i, ret_i, ret_l)
```

in the source file `C_concat.c` :

```
$ /usr/bin/genso -o lib/C_utils.so C_concat.o
```

Refer to *Oracle Database Development Guide* for the C code example.

15.1.2 Set Up the Environment

When you use the default configuration for external procedures, Oracle Database spawns `extproc` directly. You need not make configuration changes for `listener.ora` and `tnsnames.ora`. Define the environment variables to be used by external procedures in the file `extproc.ora` located at `$ORACLE_HOME/hs/admin` using this syntax:

```
SET name=value (environment_variable_name=value)
```

If the load library is a BS2000 LMS library you must set the variable `EXTPROC_DLLS` to `ANY`, For example:

```
SET EXTPROC_DLLS=ANY
```

If the load library is a POSIX shared object you can set the variable `EXTPROC_DLLS` as described in *Oracle Database Development Guide*. If you have not specified the full qualified file path for the shared object then you must set the variable `LD_LIBRARY_PATH`. Following is an example of an `extproc.ora` file for the shared object file `C_utils.so`, which resides in the `/home/oracle/lib` directory:

```
SET EXTPROC_DLLS=ONLY:C_utils.so
SET LD_LIBRARY_PATH=/home/oracle/lib
```

To change the default configuration for external procedures and have your `extproc` agent spawned by Oracle Listener, configure your `listener.ora` and `tnsnames.ora` as follows. The `listener.ora` file must have the following entries:

```
EXTPLSNR=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=ipc)(KEY=extp))))

SID_LIST_EXTPLSNR=
  (SID_LIST=
    (SID_DESC=
      (SID_NAME=ep_agt1)
      (ORACLE_HOME=/u01/app/oracbase/product/19.3.0/dbhome_1)
      (ENVS="EXTPROC_DLLS=ANY LD_LIBRARY_PATH=/home/oracle/lib")
      (ORACLE_SID=extp)
      (PROGRAM= extproc)))
```

Assign the environment variables to be used by external procedures to the parameter `ENVS`. When `extproc.ora` is in use, it precedes the same environment variables of `ENVS` in `listener.ora` file.

The `tnsnames.ora` file must have the following entry:

```
extproc_agent=
  (DESCRIPTION=
    (ADDRESS=PROTOCOL=ipc)(KEY=extp))
    (CONNECT_DATA=
      (PRESENTATION=RO)
      (SID=ep_agt1)))
```

Now, you can start the `extproc` listener.

> **See Also:**
>
> - *Oracle Database Development Guide*
> - *Oracle Database Data Cartridge Developer's Guide*
> - *Oracle Database Net Services Administrator's Guide*

15.1.3 Identify the DLL

The objects in the DLL contain the functions called as external procedures. When `extproc` is loaded, these functions are dynamically linked to the program. Use the `CREATE LIBRARY` statement to create a schema object called an alias library, which represents the DLL. The name of the DLL must match the filename specified in the value of the `extproc.ora` variable `EXTPROC_DLLS`.

```
CREATE LIBRARY [schema_name.]library_name
  {IS | AS} 'file_path'
  [AGENT 'agent_link'];
```

For the default configuration you must not specify the parameter `AGENT`. An example for a BS2000 LMS library is as follows:

```
CREATE LIBRARY C_utils AS '[$extp_userid.]C.UTILS.MODLIB';
```

An example for a POSIX shared object is:

```
CREATE LIBRARY C_utils AS '[path/]C_utils.so';
```

If you want to change the default configuration for external procedures and have your `extproc` spawned by an Oracle Listener, then you must define a database link which is used with the `AGENT` parameter of the `CREATE LIBRARY` statement.

```
CREATE DATABASE LINK agent_link USING 'extproc_agent';
CREATE OR REPLACE LIBRARY C_utils IS 'C.UTILS.MODLIB' agent
'agent_link';
```

> **See Also:**
>
> *Oracle Database Development Guide*

15.1.4 Publish External Procedures

Oracle Database can use only external procedures that are published through a call specification, which maps names, parameter types, and return types for your C external procedures to their SQL counterparts. It is written like any other PL/SQL

stored procedure except that, in its body, instead of declarations and a `BEGIN END` block, you code the `AS LANGUAGE` clause.

The following example illustrates how you can publish an External Procedure as a PL/SQL function:

```
CREATE OR REPLACE FUNCTION plsToC_concat_func (
   str1 IN VARCHAR2,
   str2 IN VARCHAR2)
RETURN VARCHAR2 AS LANGUAGE C
NAME "concat"
LIBRARY C_utils
WITH CONTEXT
PARAMETERS (
CONTEXT,
str1   STRING,
str1   INDICATOR short,
str2   STRING,
str2   INDICATOR short,
RETURN INDICATOR short,
RETURN LENGTH short,
RETURN STRING);
```

> **See Also:**
>
> *Oracle Database Development Guide*

15.1.5 Run External Procedures

Calling an external C procedure depends on the type of publishing the external procedure as a PL/SQL procedure, package, or function.

An example of how to publish the external C procedure as a PL/SQL function is described in the *Oracle Database Development Guide*.

You can also run this function within a select statement as follows:

```
select plsToC_concat_func('hello ', 'world') from DUAL;
```

If your external procedure writes messages to `STDOUT`, then you can find these messages in the `L.`*`sid`*`.EXTP.SYSOUT.`*`tsn`* file. Set the environment variable `BGJOUT` to `KEEP` to avoid the cleanup of the `L.`*`sid`*`.EXTP.SYSOUT.`*`tsn`* file.

16
Globalization Support

This chapter describes the globalization support available with Oracle Database 19c for Fujitsu BS2000, with information about the following:

- Language, Territory, and Character Set
- Supported Language Conventions
- Supported Territories
- Supported Character Sets
- Location of Message Files
- Linguistic Definitions

Character set tables and country and regional information, such as, date format, names of months, and so on, are dynamically loaded at run time. This reduces the actual storage requirements and allows new languages to be added in the future without the need to relink all applications.

The files containing character set information are created in the current BS2000 user ID. The names of these files have the following format:

```
0193NLS.LXnnnnn.NLB
```

These files are for internal use only. You should not make changes to them. If you need a character set, language, or a territory code that is not present, then contact Oracle Support Services to check for any updates.

> **Note:**
>
> User-defined character sets as documented in *Oracle Database Globalization Support Guide* are not supported in this release.

16.1 Language, Territory, and Character Set

To choose a language, a territory, and a character set that you want to work with, you must perform separate procedures for Oracle Database and the supported Oracle Database utilities. This section includes the following topics:

- Oracle Database
- Other Oracle Database Products

16.1.1 Oracle Database

For Oracle Database, the database administrator sets the `NLS_LANGUAGE` and `NLS_TERRITORY` parameters in the initialization files.

> **See Also:**
>
> *Oracle Database Globalization Support Guide* for more information

16.1.2 Other Oracle Database Products

For the supported Oracle Database products, you can choose a language, a territory, and a character set by setting the value of the `NLS_LANG` environment variable. Set this environment variable in the `ORAENV` file or in the POSIX shell as follows:

```
NLS_LANG=language_territory.characterset
```

where:

`language` is any supported language.

`territory` is any supported territory.

`characterset` is the character set required by your terminal.

For example:

```
NLS_LANG=German_Germany.D8BS2000
```

16.2 Supported Language Conventions

Oracle Database 19c for Fujitsu BS2000 provides support for language conventions, such as names of days and months, for the following languages:

- American English: `american` (default)
- Czech: `czech`
- Danish: `danish`
- Dutch: `dutch`
- Finnish: `finnish`
- French: `french`
- German: `german`
- Hungarian: `hungarian`
- Italian: `italian`
- Norwegian: `norwegian`
- Polish: `polish`
- Portuguese: `portuguese`
- Slovak: `slovak`
- Spanish: `spanish`
- Swedish: `swedish`
- Russian: `russian`
- Turkish: `turkish`

16.3 Supported Territories

Oracle Database Globalization Support provides support for territory conventions, such as start day of the week, for the following territories:

- America: `america` (default)
- Czech Republic: `czech republic`
- Denmark: `denmark`
- Finland: `finland`
- France: `france`
- Germany: `germany`
- Hungary: `hungary`
- Italy: `italy`
- The Netherlands: `the netherlands`
- Norway: `norway`
- Poland: `poland`
- Portugal: `portugal`
- Spain: `spain`
- Sweden: `sweden`
- CIS: `CIS`
- Slovakia: `slovakia`
- Turkey: `turkey`
- United Kingdom: `united kingdom`

16.4 Supported Character Sets

Oracle Database 19c supports the following character sets for servers and clients under BS2000:

Name	Description	Usage
US8BS2000	Siemens 9750-62 EBCDIC 8-bit	American
D8BS2000	Siemens 9750-62 EBCDIC 8-bit	German
F8BS2000	Siemens 9750-62 EBCDIC 8-bit	French
E8BS2000	Siemens 9750-62 EBCDIC 8-bit	Spanish
DK8BS2000	Siemens 9750-62 EBCDIC 8-bit	Danish
S8BS2000	Siemens 9750-62 EBCDIC 8-bit	Swedish

Name	Description	Usage
WE8BS2000	Siemens EBCDIC.DF.04-1 8-bit	West European (= ISO 8859/1)
CL8BS2000	Siemens EBCDIC.EHC.LC 8-bit	Latin/Cyrillic-1 (= ISO 8859/5)
WE8BS2000L5	Siemens EBCDIC.DF.04-9 8-bit	WE & Turkish (= ISO 8859/9)
EE8BS2000	Siemens EBCDIC.EHC.L2 8-bit	East European (= ISO 8859/2)
CE8BS2000	Siemens EBCDIC.DF.04-2 8-bit	Central European (= ISO 8859/2)
WE8BS2000E	Siemens EBCDIC.DF.04-F 8-bit	West European with Euro symbol (= ISO 8859/15)

The character sets WE8BS2000, CL8BS2000, WE8BS2000L5, EE8BS2000, CE8BS2000, and WE8BS2000E are the recommended database character sets. The other character sets must only be used as the client character sets.

The character set WE8BS2000E must be used as a database character set to store the euro symbol in the database or to use the euro symbol as the dual currency symbol.

In addition to these supported character sets, if you are connecting to Oracle Database installations with a non-BS2000 character set, then those servers can use any of the character sets as listed in *Oracle Database Globalization Support Guide*.

Note:

A Unicode database character set is not supported on BS2000. To store Unicode characters in the database, you must use Unicode datatypes `NCHAR`, `NVARCHAR2`, and `NCLOB`. During database creation you can specify either AL16UTF16 or UTF8 as the national character set for these data types.

See Also:

Oracle Database Globalization Support Guide for more information about Unicode support

16.5 Location of Message Files

All message files are located in `ORAMESG.LIB` in the installation user ID.

16.6 Linguistic Definitions

All the linguistic definitions listed in *Oracle Database Globalization Support Guide* except the Unicode Collation Algorithm (UCA) collations are available.

Part V
Appendices

Appendices contain supplemental material for this document.

The following topics are discussed:

- Oracle Error Messages for BS2000
- Oracle Environment Variables
- Initialization Parameters and the Parameter File
- Troubleshooting
- File Types and Names Used by Oracle

A

Oracle Error Messages for BS2000

This appendix lists error messages, causes, and corrective actions that are specific to operation of Oracle Database for Fujitsu BS2000. The messages shown in this appendix may be accompanied by additional text when displayed on screen. This text identifies the function that detected the problem, and can include internal status codes, BS2000 SOSD error code, or both. These error codes can be helpful to the Oracle Support Services Representative in determining the cause of a problem. The BS2000 SOSD error code indicates that the error originated in the operating system code. The error code is displayed in hexadecimal, and is structured as follows:

`BS2000 SOSD error 0x8`*`xxxyyyy`*` from `*`mmmmmmmm`*` : `*`text`*

Where:

- *xxx* identifies the function reporting the error.
- *yyyy* details the error. It is either an internal code of the function, or a compacted return code of a BS2000 system macro (see subsequent section).
- *mmmmmmmm* is the name of the function.
- *text*, if present, explains the error code. Often it says "`RC FROM `*`zzzzz`*` MACRO`".

A BS2000 system macro return code is condensed into the 2-byte value *yyyy* as follows:

- For system macros that return a code `bb0000aa`, *yyyy* is `bbaa`
- For I/O calls, *yyyy* is the DMS error code
- In all other cases, *yyyy* contains the right halfword of the return code of the BS2000 macro.

Sometimes, for example, in the early stages of initialization when the message components are not yet available, the Oracle Database cannot issue a regular Oracle message. If this occurs, then Oracle Database calls the ILCS task termination routine, or it issues a `TERM` macro directly, giving the message number as the user termination code. You can use this message number to find the explanation in this appendix.

ORA-05000: ORACLE termination routine called
Cause: The termination routine of the Oracle Database run-time system has been called due to a fatal error.

Action: If you do not know why the Oracle Database program terminated, or how to resolve this problem, then contact the Oracle Support Services Representative.

ORA-05001: Unsupported BS2000 Version
Cause: The active version of the BS2000 operating system is not supported by this Oracle Database release.

Action: Upgrade to a more recent BS2000 version.

ORA-05002: Fatal error: called from non-ILCS program
Cause: In a precompiler or OCI application, the Oracle Database is called from a program that does not run in an ILCS environment. The Oracle Database does not support non-ILCS programs

Action: Ensure that the application program runs in an ILCS mode. Some programming languages, for example, FOR1, PL/I, require specific options for ILCS. Refer to the Fujitsu documentation for further information.

ORA-05003: Fatal error: ILCS PCD cannot be verified
Cause: In a precompiler or OCI application, Oracle Database is called with a save area that is marked as an ILCS save area but does not point to a proper PCD (ILCS global area). The problem is either that memory has been overwritten, or that Oracle Database is called from a program that does not run in an ILCS environment. Oracle Database does not support non-ILCS programs.

Action: Ensure that the application program runs in an ILCS mode. Some programming languages, for example, FOR1, PL/I, require specific options for ILCS. Refer to the Fujitsu documentation for further information.

ORA-05004: Fatal error: stack overflow, extension failed
Cause: A call to a function required an extension of the current call stack segment. This extension failed and the corresponding ILCS routine returned the error.

Action: Ensure that the address space limit of the BS2000 user ID is sufficient and that there is no temporary memory saturation. Then re-run the program. If you need further help, then contact the Oracle Support Services Representative.

ORA-05005: Error: IT0INITS called in PROLOD
Cause: This is an internal error and should not occur.

Action: Contact the Oracle Support Services Representative.

ORA-05006: sltga already initialized
Cause: The initialization routine for the `sltga` is called more than once.

Action: Check if more than one stub modules (`PROSTUB`, `XAOSTUB`) are linked to the application.

ORA-05007: failed to load OSNTAB
Cause: This message is usually preceded by a BS2000 BLS-*nnnn* message. The most likely reason is that the `ORALOAD` library cannot be found.

Action: Contact the Database Administrator about the `ORALOAD` library. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05008: failed to load requested network driver
Cause: This message is usually preceded by a BS2000 `BLS-nnnn` message. The most likely reason is that the `ORALOAD` library cannot be found.

Action: Contact the Database Administrator about the `ORALOAD` library. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05010: bad filename length
Cause: Buffer overflow while building/translating a file name. This could be caused by specifying an excessively long file name in the `ORAENV` file.

Action: If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05011: bad file size
Cause: This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05012: bad block size
Cause: This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05013: bad filename parse
Cause: A file name being analyzed is not well-formed for Oracle Database purposes.

Action: Correct the file name and re-run the program.

ORA-05014: sfcopy: non-matching block size
Cause: In a partial database file copy, source and target file have different block sizes. This may indicate an internal error and should not normally occur.

Action: If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05015: text file open failed
Cause: An Oracle Database text or command file cannot be opened. One of the following could cause this error: the file name is wrong, the file has not been properly initialized, or the file is not accessible.

Action: Correct the problem and restart the Oracle Database. If this occurs when you issued the `STARTUP` command, then check the initialization file for the correct specification of the database files.

ORA-05016: text file close failed
Cause: Attempt to close an Oracle Database file has failed. This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05017: file open failed
Cause: An Oracle Database database file cannot be opened. Either the file name is wrong, the file has not been properly initialized, or the file is not accessible.

Action: Correct the problem and restart the Oracle Database. If this occurred when you issued the `STARTUP` command, then check the initialization file for the correct specification of the database files.

ORA-05018: file seek failed
Cause: This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05019: file write failed
Cause: An I/O error occurred while writing to an Oracle Database file.

Action: If the error cannot be identified as one caused by a disk malfunction, then either contact the System Administrator, or contact the Oracle Support Services Representative.

ORA-05020: write block outside of file
Cause: An attempt was made to write a block of an Oracle Database file that does not exist. For example, block number < 1 or > file size. This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05021: file read failed
Cause: An I/O error occurred while reading an Oracle Database file.

Action: If the error cannot be identified as one caused by a disk malfunction, then either contact the System Administrator, or contact the Oracle Support Services Representative.

ORA-05022: read block outside of file
Cause: An attempt was made to read a block of an Oracle Database file that does not exist. For example, block number < 1 or > file size. This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05023: file close failed
Cause: The attempt to close an Oracle Database file failed. This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05025: sfccf:file mismatch. Trying to reuse a file with different size
Cause: When trying to reuse a database file, the file size specified differs from the actual size of the existing file.

Action: Specify the correct file size (remember to subtract one logical block for the implicit header block), or leave the size unspecified, or use a different file name if you want to create a larger or smaller database file.

ORA-05026: file does not exist
Cause: An attempt was made to access a database file, which no longer exists.

Action: Contact the Database Administrator who may know why this error has occurred. If the Database Administrator cannot find the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05027: file does exist
Cause: When attempting to create a new file, this error occurs if the file is found and is not empty.

Action: If the error occurred in a `create database`, then retry with the `reuse` option. Otherwise use a different file name or check whether the file can be erased.

ORA-05028: file is not a dbfile
Cause: The database, or log, or control, file to be opened does not contain the proper identification for such a file.

Action: Check for wrong file specification.

ORA-05029: illegal use-option
Cause: Internal error. Function `sfccf` was called with an illegal option.

Action: Contact the Oracle Support Services Representative.

ORA-05030: SID not defined
Cause: When the system id was required, typically, to substitute the "?" in names, for example, in file names set by the initialization file, it was not yet defined. This could be caused by a missing `ORAENV` file or a missing `ORASID` in that file.

Action: Ensure that the `ORAENV` file definition is correct and re-run the program.

ORA-05031: SID translation failure
Cause: The system id is syntactically incorrect.

Action: Ensure that the `ORASID` definition is correct and re-run the program.

ORA-05032: bad name parse
Cause: The translation of a file name, or other name containing variable parts, failed. The error may be caused by a wrong specification in the `ORAENV` file.

Action: Ensure that the `ORAENV` variable assignments are correct. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05033: bad environment values
Cause: One or more of the values specified in the `ORAENV` file are invalid.

Action: Ensure that you specified legal values in the `ORAENV` file, refer to the Oracle Environment Variables in this guide for further information.

ORA-05034: bad seal
Cause: Internal error. An internal file control structure is found to be corrupt.

Action: Contact the Oracle Support Services Representative.

ORA-05035: host command not executed
Cause: A BS2000 command, argument of a `HOST` or `#HOST` command, is invalid or too long.

Action: Enter a valid `HOST` command.

ORA-05036: bad user id (*length*)
Cause: Internal buffer overflow while building a file name from variable components.

Action: Ensure that the `ORAUID` value specified in the `ORAENV` file is correct. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05037: /CANCEL command not executed
Cause: A background job could not be canceled. The background task may have already been terminated.

Action: If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05038: SID has illegal length
Cause: The system identifier specified in either the `ORAENV` file or as part of a connect string exceeds 4 characters in length.

Action: Specify a correct value.

ORA-05039: Recursive entry to ssodrv
Cause: Oracle Database kernel has been reentered at the top. This should not happen.

Action: Ensure that the user program does not incorrectly call Oracle Database functions from within an interrupt handling routine (signal routine, contingency). If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05040: no more dynamic memory
Cause: Request memory failed in file-management components. This is probably caused by a user address space that is too small.

Action: Ensure that the address space limit of the BS2000 user ID is sufficient and that there is no temporary memory saturation. Then re-run the program. If you need further help, then contact the Oracle Support Services Representative.

ORA-05046: Archive control string error
Cause: The archive file name or control parameters are incorrect.

Action: Correct the parameters.

ORA-05050: PGA (fixed part) could not be allocated
Cause: Probable operating system error or internal error.

Action: Contact the Oracle Support Services Representative.

ORA-05051: cannot allocate var. PGA
Cause: During creation of the PGA, required dynamic memory could not be allocated.

Action: Verify that the user address space is large enough and that if an application program produced the error, the program is not consuming excessive memory. Otherwise contact the Oracle Support Services Representative.

ORA-05052: error deleting var. PGA
Cause: During deletion of the PGA, dynamic memory could not be released. This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05053: invalid or missing PGA_BASE
Cause: An invalid value for the `PGA_BASE` parameter has been specified in the `ORAENV` file.

Action: Use the default value for the PGA_SIZE environment variable. If this does not solve the problem, then contact the Oracle Support Services Representative.

ORA-05054: invalid or missing PGA_SIZE
Cause: An invalid value for the PGA_SIZE environment variable has been specified in the ORAENV file. You should never need to change the default value for the PGA_SIZE environment variable.

Action: Use the default value for the PGA_SIZE environment variable. If this does not solve the problem, then contact the Oracle Support Services Representative.

ORA-05055: address range for PGA (fixed part) is not free
Cause: The address range described by the PGA_BASE and PGA_SIZE ORAENV variables is not available for allocation. This may be due to overlapping PGA, SGA, and KERNEL areas, or to an application program, which has occupied memory in this area. If you did not specify a value for PGA_BASE, the default may be inappropriate for the case.

Action: Refer to Memory Architecture for further information.

ORA-05056: no more context space
Cause: During processing of a SQL statement, dynamic memory could not be allocated. This could happen when very complex requests are being processed and there is not enough memory available.

Action: Verify that the user address space is large enough and that the application program, if the error occurred when you were using an application program, is not using excessive memory. Otherwise, contact the Oracle Support Services Representative.

ORA-05058: assert failed: SGA not mapped
Cause: This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05059: assert failed: not in kernel
Cause: This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05060: SGA not created
Cause: After you run the STARTUP command, the SGA shared memory pool could not be created.

Action: Verify that you are not trying to start the database while it is running and that the database system id is not being used for two different databases. Otherwise, contact the Oracle Support Services Representative.

ORA-05061: SGA attach failed
Cause: Connection to the SGA shared memory pool could not be established. This may have happened if you used the wrong system id, or if the database you expected to be running is not running.

Action: Verify that it is not one of the preceding causes (check with the Database Administrator). Otherwise, contact the Oracle Support Services Representative.

ORA-05063: SGA base invalid
Cause: An invalid value has been specified for the SGA_BASE parameter in the ORAENV file.

Action: This value is not normally needed. If specified, it must be a value giving the full virtual address for the SGA memory pool. Correct the value and run the STARTUP command.

ORA-05064: cannot allocate SGA
Cause: After creating the memory pool, the REQMP to allocate the space failed. This might be an operating system error.

Action: If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05065: SGA not deleted
Cause: When attempting to detach from the SGA, the DISMP system macro returned an error.

Action: If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05066: SGA address space conflict
Cause: The SGA cannot be placed at the requested address range, because the range is already partly used. The SGA start address is defined by the ORAENV variable, SGA_BASE; its size is determined by various initialization file parameters such as processes, buffers, and so on.

Action: Refer to “Memory Architecture” and adjust the relevant initialization file and ORAENV variables. Check the address space limit of the DBA user ID. Contact the System Administrator to find out about shared subsystems and their placement in the address space. Ensure that you do not overlap with the Oracle Database kernel.

ORA-05067: SGA: address space saturation
Cause: When the SGA is being allocated, the operating system reported that the virtual address space is saturated.

Action: Contact the System Administrator about paging area size and current overall system load.

ORA-05068 SGA still active, should not be
Cause: When the SGA is being created during startup, it is found that the SGA memory pool is still in use, although the databases should be shut down. This may be caused by a hanging user task, or a network server task.

Action: Check for such hanging tasks. Cancel these tasks, and then restart the database.

ORA-05069: Unexpected SGA memory pool problem
Cause: The ENAMP macro returned an unexpected error code.

Action: Contact the Oracle Support Services Representative.

ORA-05070: cannot enable TPA ser.item
Cause: Probable operating system error.

Action: Contact the Oracle Support Services Representative.

ORA-05071: cannot ENQ on TPA ser.item
Cause: Probable operating system error.

Action: Contact the Oracle Support Services Representative.

ORA-05072: cannot enable post/wait item
Cause: Probable operating system error.

Action: Contact the Oracle Support Services Representative.

ORA-05073: error in post
Cause: An inter-process communication operation failed.

Action: Check that the database and all required background tasks are running correctly. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05074: error in wait
Cause: An inter-process communication operation failed.

Action: Check that the database and all required background tasks are running correctly. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05075: error in task table manager
Cause: Internal error.

Action: Contact the Oracle Support Services Representative.

ORA-05076: error setting spid
Cause: Probable operating system error.

Action: Contact the Oracle Support Services Representative.

ORA-05077: cannot enable HIA event
Cause: Probable operating system error. The HIA (Here I Am) event item is used to check the successful start of an Oracle Database process.

Action: Contact the Oracle Support Services Representative.

ORA-05078: create process failure
Cause: When you issued the `STARTUP` command, a background job could not be started successfully.

Action: Check for the job scheduling problems and that any `BGJPAR` entry in the `ORAENV` file is correct. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05079: internal asynchronous IO error
Cause: This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05107: POSIX environment variable <variablename> not defined
Cause: The specified environment variable is not defined.

Action: Define and export the requested variable in your profile.

ORA-05108: failed to process BS2000 command <bs2-command>
Cause: The BS2000 command processor cannot execute the command.

Action: Test the logged command in the POSIX shell using the POSIX command, `bs2cmd`.

ORA-05109: failed to initialize environment for POSIX
Cause: An application running under the POSIX shell cannot create links to required files in the BS2000 file system.

Action: Check if the environment variables required for Oracle Database applications under POSIX are set properly.

ORA-05110: cannot attach to memory pool
Cause: Invalid pool ID parameter `xxx_MPID` or operating system error.

Action: Check the `ORAENV` parameter `xxx_MPID`, at most 4 characters of the set [A...Z], [0...9]), or contact the Oracle Support Services Representative.

ORA-05111: error attaching to memory pool
Cause: This error mostly occurs due to an address space conflict. The two low significant bytes of the SOSD error show the return code of the ENAMP macro. For example, 1804.

Action: Refer to the manual *Executive Macros* of the Fujitsu BS2000 operating system released on the Fujitsu manual server.
This manual contains a description of the SOSD error. This problem usually can be resolved by using a higher base address.

ORA-05112: error creating memory pool
Cause: This error mostly occurs due to an address space conflict. The two low significant bytes of the SOSD error show the return code of the ENAMP macro. For example, 1804.

Action: Refer to the manual *Executive Macros* of the Fujitsu BS2000 operating system released on the Fujitsu manual server.
This manual contains a description of the SOSD error. This problem usually can be resolved by using a higher base address

ORA-05114: bad pool base
Cause: An invalid value for the base address parameter of the shared pool, that is, `CLN_BASE, COM_BASE`, and so on has been specified in the `ORAENV` file.

Action: If this value is specified, it must be a value giving the full virtual address for the base address of a memory pool. Correct the value and restart the database.

ORA-05116: cannot load shared code into pool
Cause: Shared code could not be loaded into the specified memory pool. The two low significant bytes of the SOSD error show the main return code of the BIND macro, for example, x'0198'.

Action: Refer to the manual *Executive Macros* of the Fujitsu BS2000 operating system released on the Fujitsu manual server for a description of the SOSD error.

ORA-05117: cannot attach to socket subsystem
Cause: An application could not be bound to the sockets subsystem. Generally this message is preceded by a BLS-nnnn message from the operating system.

Action: Use the BS2000 command `'SHOW-SUBSYSTEM-STATUS SOC6'` to verify that the sockets subsystem SOC6 is created.

ORA-05118: ORACLE PCD slot not accessible
Cause: The current task is trying to attach to the ORACLE PCD slot but cannot find this slot.
This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05119: module verification failed
Cause: The version of the shared loaded module does not match the version of the connection module on the user side.

Action: Mostly this error occurs with user applications. In this case re-link the application with the Oracle Database release in use.

ORA-05120: waiting for shared module to be loaded timed out
Cause: This is an internal error and should not normally occur.

Action: Set the environment Variable `OSDDBG=INI` and re-start the program. The error and the involved memory pool name will be logged. Ask your BS2000 system administrator if the associated memory pool is locked by a foreign task or user.

ORA-05121: waiting for initialization of shared module timed out
Cause: This is an internal error and should not normally occur.

Action: Set the environment Variable `OSDDBG=INI` and re-start the program. The error and the involved memory pool name will be logged. Ask your BS2000 system administrator if the associated memory pool is locked by a foreign task or user.

ORA-05126: Missing IT0PCD address
Cause: The `ILCS` run-time link-library is probably missing.

Action: Contact the System Administrator.

ORA-05161: TCP/IP can't perform asynchronous test on break socket.
Cause: Select on break socket failed.

Action: Contact the System Administrator about TCP/IP networking problems. If the error persists, contact the Oracle Support Services Representative.

ORA-05165: function not supported
Cause: Either Oracle Database or BS2000 does not support this function.

Action: None.

ORA-05167: Defect in data buffer
Cause: This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05170: SID not defined (ORAENV file missing?)
Cause: The system identifier, data base name, is not defined when needed during Oracle Database program initialization. A missing `ORAENV` file or a missing `ORASID` entry in that file could cause this error.

Action: Ensure that the `ORAENV` file definition is correct and re-run the program.

ORA-05173: bad kernel size
Cause: An invalid value for the `KNL_SIZE` parameter has been specified in the `ORAENV` file.

Action: You should not normally need to specify this variable, as the default value is correct. Contact the Oracle Support Services Representative.

ORA-05174: bad kernel base
Cause: An invalid value for the `KNL_BASE` parameter has been specified in the `ORAENV` file.

Action: If this value is specified, then it must be a value giving the full virtual address for the kernel memory pool. Correct the value and restart the database.

ORA-05175: Kernel address space conflict
Cause: The Oracle Database kernel cannot be placed at the requested address range, because the range is already used. The kernel start address is defined by the `ORAENV` parameter, `KNL_BASE`.

Action: Refer to Memory Architecture and adjust the relevant initialization file and `ORAENV` parameters. Check the address space limit of the DBA user ID. Contact the System Administrator to learn about shared subsystems and their placement in the address space.

ORA-05176: Kernel: address space saturation
Cause: When the Oracle Database kernel memory pool was being allocated, the operating system signalled that the virtual address space is currently saturated.

Action: Contact the System Administrator about paging area size and current overall system load.

ORA-05177: Unexpected Kernel memory pool problem
Cause: The `ENAMP` macro returned an unexpected error code.

Action: Refer to the `ENAMP` macro description in the BS2000 documentation for possible reasons. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05178: Kernel module not yet initialized
Cause: The current task is trying to attach to an Oracle Database kernel which is not yet completely initialized. This can only happen if you try to connect to a database, which is just being started.

Action: Retry after a while. Remember that it may take a few minutes until a database is fully running and ready for the users. If the error persists, then check this issue with the Database Administrator.

ORA-05181: load/init problem with PRO/OCI interface
Cause: The user-side stub module could not load the PRO/OCI module. In this case, the message is usually preceded by a BS2000 BLS-*nnnn* message, or the loaded module is incompatible with the version of the stub module.

Action: Ensure that the `ORALOAD` link name exists and points to the current `ORALOAD` library. Re-link the application with the current link libraries.

ORA-05191: symbol translation error for kernel memory pool
Cause: The logical name translation for the kernel memory pool failed. Normally, this indicates an invalid system id, `ORASID` in the `ORAENV` file.

Action: Ensure that the `ORAENV` file definition is correct. Otherwise, contact the Oracle Support Services Representative.

ORA-05192: cannot create/attach kernel memory pool
Cause: The memory pool for the Oracle Database kernel code could not be enabled. In a user program, a possible cause is that the user program already allocates part of the address range needed for the memory pool.

Action: Ensure that the user program does not request storage excessively, and that any `SGA_BASE` and `KNL_BASE` parameters in the `ORAENV` file are consistent. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05193: Symbol translation error for kernel module or load library
Cause: The logical-name translation for the kernel module or load library failed. This is an internal error and should not normally occur.

Action: Contact the Oracle Support Services Representative.

ORA-05194: cannot load kernel
Cause: The kernel could not be loaded into the kernel memory pool. In most cases, this message is preceded by a `BLS-`*nnn* message from the operating system.

Action: Ensure that the `ORALOAD` link name identifies the correct `ORALOAD` library, and that the `ORAENV` variable, `KNL_MODULE`, names one of the possible kernels. Then re-issue the `STARTUP` command. If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05195: bad or missing kernel connector
Cause: The loaded kernel could not verify its user-side connector module. This can occur if you use an incorrect kernel version.

Action: If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05198: associated internal OSD error code %d
Cause: This message precedes ORA-05199, if there is more information available. The first 4 hexadecimal digits can often identify the module, and the last 4 hexadecimal digits are usually a condensed version of an associated system macro code. This code can be helpful in diagnosing the problem.

Action: If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

ORA-05199: ORACLE ABNORMAL EXIT
Cause: A fatal error occurred, which prevents continuation of execution. In many cases, a preceding message explains the error. The system causes the program execution to stop (`TERM ABNORMAL` with `DUMP` is displayed).

Action: If you cannot identify the cause of the problem, then contact the Oracle Support Services Representative.

B
Oracle Environment Variables

This appendix describes variables that can be specified in the `ORAENV` file, the structured system variable `SYSPOSIX`, or the POSIX shell. Oracle parameters, such as `ORACLE_SID` and `NLS_LANG`, may be specified in the `ORAENV` file, system variable `SYSPOSIX`, or POSIX shell. If you use a `ORAENV` file, then you must follow the `ORAENV` rules for specifying environment variables as described in the following sections. If you use the system variable `SYSPOSIX`, then you must follow the rules to set a `SDF-P` variable. For example, if the variable name contains an underscore '_' , then you must use a hyphen '-' instead of the underscore. In the POSIX shell, you must follow the UNIX rules to set and export the environment variables.

This appendix contains the following topics:

> **See Also:**
>
> "Oracle Envrionment Variables" for more details

B.1 ORAENV Rules

Consider the following general rules when you create or modify `ORAENV` files:

- All lines which begin with a slash or asterisk (/ or *) are ignored.
- All variable names must be written in uppercase.
- Spaces must not be included immediately before and after the equals sign (=).
- Do not enclose values in quotation marks unless you want the quotation marks to be part of the value.
- Errors in variable names are not recognized. This means that the value of any variable whose name is typed incorrectly is not modified.
- There is only limited checking of variable assignments. An incorrect value may generate an error message, but may also be interpreted as a null value.
- When variable assignments refer to other variables, BS2000 command file substitution syntax applies. Substitution takes place when the variable is stored in the local environment.

  For example:

```
ORAUID=$ORACINST
SQLPATH=&ORAUID..RDBMS.ADMIN
```

assigns the value `$ORACINST.RDBMS.ADMIN` to the variable `SQLPATH`. If `ORAUID` is changed, then `SQLPATH` automatically reflects the new value.

- The sequence of items in the `ORAENV` file is not generally significant. If an item occurs more than once, then the first occurrence is used.
- If a value is not specified for a variable, then the default value is used, if it exists.

B.2 Built-in Variables

The following variables are always defined, and may be referenced in other variable assignments:

- LOGNAME
- ORAUID
- PGM
- TERM
- TSN

B.2.1 LOGNAME

The `LOGNAME` variable always contains the current BS2000 user ID. Do not alter the value of this variable by assigning a different value to it in the `ORAENV` file.

B.2.2 ORAUID

This variable specifies the BS2000 user ID where the Oracle Database programs, installation, and demonstration files are installed. The initial value is derived from the `ORALOAD` link name (the user ID part of the `ORALOAD` library name). This value is usually correct, but if necessary, you can override it by assigning a different value to it in the `ORAENV` file.

Format: `ORAUID=`*`$userid`* or `ORAUID=/BS2/`*`$userid`*

B.2.3 PGM

The `PGM` variable contains the program name specified in the BS2000 command `START-EXECUTABLE`. You cannot alter the value of this variable by assigning a different value to it in the `ORAENV` file.

B.2.4 TERM

The `TERM` variable contains the terminal type. The default value is SNI9750. This default value is usually correct, but if necessary, you can override it by assigning a different value to it in the `ORAENV` file.

B.2.5 TSN

The `TSN` variable contains the task sequence number of the current task. You cannot alter the value of this variable by assigning a different value to it in the `ORAENV` file.

B.3 General Variables

The following variables are for general use by Oracle DBAs and users:

- CLN_BASE
- CLN_MPID
- CLN_SCOPE
- EXP_CLIB_FILE_IO
- IMP_CLIB_FILE_IO
- NLS_LANG
- OPS_JID
- ORASID
- PRINTPAR
- SQLPATH
- SSSIDPWF

B.3.1 CLN_BASE

This variable specifies the address of the shared memory pool for the client-side shared code of Oracle Database, which is used by customer written database applications.

Format:

`CLN_BASE=`*`address`*

Default:

200M

B.3.2 CLN_MPID

This variable specifies the identification of the shared memory pool for the client-side shared code of Oracle Database, which is used by customer written database applications.

Format:

`CLN_MPID=`*`sid`*

Default:

`CLN_MPID=`*`&ORASID`*

B.3.3 CLN_SCOPE

This variable specifies the scope of the shared memory pool for the client-side shared code of Oracle Database, which is used by customer written database applications. The valid values are:

Value	Description
T	The use of the memory pool is limited to the calling task.
U	All Oracle Database client tasks within the same BS2000 user ID as the calling task are participants of the shared memory pool.
G	All Oracle Database client tasks in the system are participants.

Format:

```
CLN_SCOPE={T|U|G}
```

Default:

```
CLN_SCOPE=G
```

B.3.4 EXP_CLIB_FILE_IO

This variable should be set to `FALSE` when you use the `Export` utility to overcome a problem with the C library functions, when an export file is written to tape.

Format: `EXP_CLIB_FILE_IO=FALSE`

Default: `EXP_CLIB_FILE_IO=TRUE`

B.3.5 IMP_CLIB_FILE_IO

This variable should be set to `FALSE` when you use the `Import` utility to overcome a problem with the C library functions, when an import file is read from tape.

Format: `IMP_CLIB_FILE_IO=FALSE`

Default: `IMP_CLIB_FILE_IO=TRUE`

B.3.6 NLS_LANG

This variable specifies the language, territory, and character set. For example:

```
NLS_LANG=GERMAN_GERMANY.D8BS2000
```

Format: `NLS_LANG=`*`language_territory.character-set`*

Default: `NLS_LANG=AMERICAN_AMERICA.WE8BS2000`

B.3.7 OPS_JID

This variable is used for concatenation with the `OS_AUTHENT_PREFIX`, refer to the initialization parameter. The default value concatenates the value of the parameter `OS_AUTHENT_PREFIX` with the BS2000 user ID. Using `OPS_JID`, you can specify that the BS2000 jobname is used instead. This is useful when many users are sharing a single BS2000 user ID.

Format: `OPS_JID=`*`userid/jobname`*

Default: *`userid`*

B.3.8 ORASID

This variable defines the database that is used if no database identification is given at connect time.This variable is a synonym of the `ORACLE_SID` variable.

Format: `ORASID=`*`sid`* (*`sid`* is a characterstring where 1 <= length <= 4)

> **Note:**
>
> Oracle recommends that you use the `ORACLE_SID` variable.

B.3.9 PRINTPAR

This variable specifies optional variables for the `/PRINT` command internally issued by the `SPOOL OUT` statement in SQL*Plus. Using this variable, the user can modify the spooled job, and, for example, route the job to a remote printer, add print options for laser printers, and so on. The BS2000 `/PRINT` command for spool files is issued as follows:

`/PRINT` *`temporary_spoolfile`*`,&PRINTPAR`

Format: `PRINTPAR=`*`print-options`*

B.3.10 SQLPATH

This variable specifies a path where SQL*Plus looks for SQL scripts. Elements of the path are separated by semicolons (;). For example:

```
SQLPATH=PRIVATE;$ORACINST
```

This assignment causes SQL*Plus to look for filename.SQL, then for `PRIVATE.`*`filename`*`.SQL`, and finally for `$ORACINST.filename.SQL`.

Format: `SQLPATH=`*`search-path`*

Default: `SQLPATH=&ORACLE_HOME/rdbms/admin;&ORAUID..RDBMS.DEMO`

B.3.11 SSSIDPWF

This variable specifies the password file.

Format: `SSSIDPWF=`*`password-file`*

> **See Also:**
>
> "Administering Oracle Database " for more information

B.4 DBA Startup Variables

The following variables are used during database and network startup. They supplement (and in some cases provide defaults for) variables specified in the initialization file.

To ensure that the variables are consistent, Oracle recommends that database startup and shutdown, background tasks, and server tasks refer to the same `ORAENV` file.

> **Note:**
>
> The default values listed in the following sections are built-in defaults, some of them are overridden by settings in the shipped `DEMO.P.ORAENV`.

The following DBA startup variables include:

- Address and Size Specification
- BGJPAR
- BGJ_PROCEDURE
- BGJPRC_UID / BGJPRC_SID
- BGJ_LOG_JOBSTART
- sid_BGJPAR
- sid_USER
- user_ACCOUNT/ user_PASSWORD
- COM_MPID
- COM_BASE
- COM_SCOPE
- JOBID
- KNL_BASE
- ORACLE_HOME
- PGA_BASE
- PGA_SIZE
- SF_PBLKSIZE
- SGA_BASE

B.4.1 Address and Size Specification

Several variables described in this section define memory addresses and sizes. The notations used to specify these items are as follows:

- A number with no modifiers is interpreted as a decimal number
- A number followed by K or M is interpreted as a decimal number multiplied by 1024 or 1048576 (1024*1024) respectively

- A number enclosed in single quotation marks and preceded by the letter X is interpreted as a hexadecimal number

For example, the following example sets the KNL_BASE variable to 8 MB:

```
KNL_BASE=8M
KNL_BASE=8388608
KNL_BASE=X'800000'
```

B.4.2 BGJPAR

This variable specifies the parameters for the BS2000 command ENTER-PROCEDURE, which is used for starting BS2000 jobs for the Oracle Database background and server processes. You can specify most of the parameters that are allowed for the BS2000 command ENTER-PROCEDURE in this variable. The ENTER-PROCEDURE command is used to submit jobs as follows:

.jobname ENTER-PROCEDURE *jobfile*,&BGJPAR

Format: BGJPAR=*parameters*

> **Note:**
>
> The BGJPAR variable is set up by the installation procedure.

B.4.3 BGJ_PROCEDURE

This variable specifies the name of the ENTER-PROCEDURE command for starting background jobs.

Format: BGJ_PROCEDURE=*filename*

Default: BGJ_PROCEDURE=(&ORAUID..ORALOAD.LIB,ENTER.PRC)

B.4.4 BGJPRC_UID / BGJPRC_SID

These variables specify the user ID and *sid* of the file for the background enter jobs. If you want to use a special enter job file, then the parameters must be set to the desired *userid* and *sid*.

Format:

```
BGJPRC_UID=$userid
BGJPRC_SID=sid
```

B.4.5 BGJ_LOG_JOBSTART

This variable specifies whether the operating system message that a new job was accepted should be logged on SYSOUT or not.

Format: BGJ_LOG_JOBSTART=Y/N

Default: BGJ_LOG_JOBSTART=N

B.4.6 sid_BGJPAR

This variable specifies the parameters, which are used by the ENTER-PROCEDURE command to start a server process for the instance specified by *SID*.

Format: *sid*_BGJPAR=*parameters*

Syntax: *sid* is a string of a maximum of 4 alphanumeric characters

parameters are the parameters for the ENTER-PROCEDURE command as described in the *BS2000 Commands manual*.

B.4.7 sid_USER

This variable specifies the BS2000 user ID where the instance identified by *sid* resides.

Format: *sid*_USER=*userid*

Syntax: *sid* is a string of a maximum of 4 alphanumeric characters

userid is a string of a maximum of 8 alphanumeric characters, which follows the naming rules of a BS2000 user ID.

B.4.8 user_ACCOUNT/ user_PASSWORD

*user*_ACCOUNT or *user*_PASSWORD define credentials of a BS2000 user ID, which is used by the ENTER-PROCEDURE command to start a process.

Format: *user*_ACCOUNT=*account*

*user*_PASSWORD=*password*

Syntax: *user* is a string of a maximum of 8 alphanumeric characters, which follows the rules of a BS2000 user ID and must match a BS2000 user ID defined by the parameter *sid*_USER.

account is a string of a maximum of 8 alphanumeric characters, which follows the naming rules for a BS2000 account number.

password is a string of a maximum of 8 alphanumeric characters, which follows the naming rules for a BS2000 password.

B.4.9 COM_MPID

This parameter specifies the identification of the shared code pool of the Oracle database instance for some common Oracle Database software components.

Format: COM_MPID=*sid*

Default: COM_MPID=*&ORASID*

B.4.10 COM_BASE

This parameter specifies the address of the shared code pool of the Oracle database instance for some common Oracle Database software components.

Format: `COM_BASE=address`

Default: Release Dependent

B.4.11 COM_SCOPE

This variable specifies the scope of the shared memory pool for the server-side shared code of Oracle Database

The valid values are:

T - The use of the memory pool is limited to the calling task.

U - All Oracle Database tasks within the same BS2000 user ID as the calling task are participants of the shared memory pool.

G - All Oracle Database tasks in the system are participants.

Format: `COM_SCOPE={T|U|G}`

Default: `COM_SCOPE=G`

B.4.12 JOBID

This variable is used internally in identifying the background tasks and generating task-specific names. You must never specify values for this variable.

B.4.13 KNL_BASE

This variable specifies the base address of the shared memory pool for the Oracle Database kernel module. This must be an integral number of megabytes.

Format: `KNL_BASE=address`

Default: Release Dependent

B.4.14 ORACLE_HOME

The Oracle home directory is the directory in the POSIX file system, which contains the installation of the software for a particular Oracle product.

Format: `ORACLE_HOME=/path-name`

B.4.15 PGA_BASE

This variable specifies the base address of the fixed part of the PGA. The PGA is task-specific, but must be located at a fixed memory address so that the kernel can access it. The base address must lie on a 64 KB boundary.

Format: `PGA_BASE=address`

Default: Release Dependent

B.4.16 PGA_SIZE

This variable specifies the size of the fixed part of the PGA. This variable must not be changed from its default value.

Format: `PGA_SIZE=size`

Default: Release Dependent

B.4.17 SF_PBLKSIZE

This variable specifies the physical blocksize of redo log files.

Format: `SF_PBLKSIZE=2K|4K`

Default: `2 KB`

> **Note:**
>
> This variable cannot be changed after database creation.

B.4.18 SGA_BASE

This variable specifies the base address of the shared memory pool for the SGA of an Oracle Database instance. The base address must lie on a megabyte boundary.

Format: `SGA_BASE=address`

Default: Release Dependent

> **Note:**
>
> There is no corresponding `SGA_SIZE` variable; the size of the SGA memory pool is calculated when the database is started.

B.5 Oracle Net Services Variables

The following are the Oracle Net Services variables:

- DEFAULT_CONNECTION
- BREAK_HANDLING
- TNS_ADMIN
- TNS_BEQ_TIMEOUT
- TNS_UPDATE_IPNODE
- TNS_DH_TIMEOUT
- NT_IPC_PROTOCOL_UNIX

B.5.1 DEFAULT_CONNECTION

This variable provides a default host string for connect requests where no host string is specified. If you connect to the same database always, then it may be convenient to

specify this variable. This value must contain everything you would otherwise specify after the "@" character. This variable is a synonym of the TWO_TASK variable.

Format: DEFAULT_CONNECTION=*host-string*

Example:

```
DEFAULT_CONNECTION=TNS:
(DESCRIPTION=
(ADDRESS=
(PROTOCOL=TCP)
(HOST=MADRID)
(PORT=1521))
(CONNECT_DATA=
(SERVICE_NAME=PROD)))
```

B.5.2 BREAK_HANDLING

This variable deactivates the signal routine for user interrupts, which sends a break over the network. An interrupt can be released by pressing the [K2] key.

Format:

```
BREAK_HANDLING=ON|OFF
```

Default:

```
BREAK_HANDLING=ON
```

B.5.3 TNS_ADMIN

This variable specifies the path to the Oracle Net Services configuration files, for example, LISTENER.ORA, TNSNAMES.ORA, and SQLNET.ORA. The configuration files can be stored in the BS2000 DMS or POSIX file system. Depending on the target file system, the path can be a BS2000 user ID or a POSIX directory.

If TNS_ADMIN is not defined, then the default search path is NETWORK.ADMIN, when the program runs in the BS2000 environment.

When the program runs in the POSIX program environment, the default search path for TNS_ADMIN is $ORACLE_HOME/network/admin.

Format: TNS_ADMIN=$*userid*|*/posix_path*

B.5.4 TNS_BEQ_TIMEOUT

This variable specifies the time a parent process waits until it establishes a connection to a child process.

Format: TNS_BEQ_TIMEOUT=*lifetime* (in seconds)

Default: TNS_BEQ_TIMEOUT=180

B.5.5 TNS_UPDATE_IPNODE

This variable forces the Oracle Net software to change the server's IP-Node name to an IP-Node address.

Format: `TNS_UPDATE_IPNODE=TRUE/FALSE`

Default: `TNS_UPDATE_IPNODE=FALSE`

B.5.6 TNS_DH_TIMEOUT

If a listener has accepted a connection request which must be handed off to a server, then this variable specifies the time the listener waits for a response from the server.

Format: `TNS_DH_TIMEOUT=sec`

Default: `TNS_DH_TIMEOUT=10`

B.5.7 NT_IPC_PROTOCOL_UNIX

Specifies whether to use the POSIX sockets protocol `PF_UNIX` for local IPC communication or the BS2000 sockets protocol `PF_ISO`.

Format: `NT_IPC_PROTOCOL_UNIX={TRUE|FALSE}`

Default: `NT_IPC_PROTOCOL_UNIX=FALSE`

> **Note:**
>
> There are some differences in the BEQ adapter when using POSIX sockets. A client using BS2000 sockets with `PF_ISO` for IPC cannot connect through IPC or BEQ adapter to a server that uses the `PF_UNIX` protocol.

C

Initialization Parameters and the Parameter File

Every time SQL*Plus starts an Oracle Database instance, it uses a set of parameters which specify the characteristics of the instance's operation. These parameters are kept in a file, typically named *sid*.DBS.INIT.ORA.This topic lists unsupported parameters, and lists other parameters that you may need to change to customize the Oracle Database for the system.

This topic contains the following sections:

- Example Parameter File
- Unsupported Parameters
- Additional Notes on Initialization Parameters

> **See Also:**
>
> *Oracle Database Reference* for general descriptions of the parameters

C.1 Example Parameter File

The $ORACINST.DEMO.DBS.INIT.ORA parameter file is created during Oracle Database installation. You can copy and edit this as a text file.

C.2 Unsupported Parameters

The following initialization file parameters, described in the generic documentation are not supported in Oracle Database 19c for Fujitsu BS2000:

- MAX_DUMP_FILE_SIZE
- OS_ROLES
- AUDIT_SYSLOG_LEVEL
- MEMORY_MAX_TARGET
- MEMORY_TARGET

Specifying these parameters in the initialization file results in an Oracle Database error during startup. The workaround is to remove such lines from the file.

C.3 Additional Notes on Initialization Parameters

This section contains additional information about the following initialization parameters:

- BACKGROUND_DUMP_DEST

- USER_DUMP_DEST
- AUDIT_FILE_DEST
- DB_BLOCK_SIZE
- DB_FILE_MULTIBLOCK_READ_COUNT
- DB_FILES
- LOCK_SGA
- SGA_MAX_SIZE
- LOG_ARCHIVE_DEST

C.3.1 BACKGROUND_DUMP_DEST

This parameter specifies the path name (directory or prefix), where debugging trace files for the background processes, such as, `LGWR`, `DBWn`, and so on are written during Oracle operations. Furthermore, it specifies the path name for the alert file. The default value for this parameter is the current BS2000 user ID of the Oracle background processes. You can specify a prefix for the trace and alert files in the following format:

```
BACKGROUND_DUMP_DEST=BDD
```

You can also specify a POSIX directory for this parameter.

> **Note:**
>
> The `BACKGROUND_DUMP_DEST` parameter is deprecated in Oracle Database 12*c* Release 1.
>
> This parameter is ignored by the new diagnosability infrastructure introduced in Oracle Database 11*g*, which places trace and core files in a location controlled by the `DIAGNOSTIC_DEST` initialization parameter.

C.3.2 USER_DUMP_DEST

This parameter specifies the path name (directory or prefix), where the server writes the debugging trace files on behalf of a user process. The default value for this parameter is the current BS2000 user ID of the Oracle Database processes.

You can specify a prefix for the trace files as follows:

```
USER_DUMP_DEST=UDD
```

You can also specify a POSIX directory for this parameter.

> **Note:**
>
> The `USER_DUMP_DEST` parameter is deprecated in Oracle Database 12*c* .
>
> This parameter is ignored by the new diagnosability infrastructure introduced in Oracle Database 11*g*, which places trace and core files in a location controlled by the `DIAGNOSTIC_DEST` initialization parameter.

C.3.3 AUDIT_FILE_DEST

This parameter specifies the operating system directory into which the audit trail is written when the `AUDIT_TRAIL` initialization parameter is set to `os`, `xml`, or `xml,extended`. You must specify a POSIX directory for this parameter. The first default value for this parameter is `ORACLE_BASE/admin/ORACLE_SID/adump`. The second default value, which is used if the first default value does not exist or is unusable, is `ORACLE_HOME/rdbms/audit`. Remember that regardless of whether database auditing is enabled, Oracle Database on Fujitsu BS2000 always records some database-related actions into the operating system audit file, such as, instance startup, shutdown, and connections with administrator privileges.

> **Note:**
>
> In an Oracle Database that has been migrated to unified auditing, the setting of this parameter has no effect.

C.3.4 DB_BLOCK_SIZE

This parameter can have one of the following values:

- 2K, 4K, 6K, 8K, 16K, 32K, if you use BS2000 2K pubset format.
- 4K, 8K, 16K, 32K, if you use BS2000 4K pubset format.

C.3.5 DB_FILE_MULTIBLOCK_READ_COUNT

The maximum value of this parameter is `128K/DB_BLOCK_SIZE`, which in most cases is also the recommended value. Setting this parameter beyond this limit has no effect.

C.3.6 DB_FILES

The maximum value for this parameter is 65533 for BS2000.

C.3.7 LOCK_SGA

This parameter is ignored in Oracle Database 12*c* for Fujitsu BS2000. Buffers in the SGA are page-fixed only during I/O operations. Otherwise, the SGA on BS2000 is pageable.

C.3.8 SGA_MAX_SIZE

This parameter should not be specified for Fujitsu BS2000. Because the SGA is not permanently page-fixed as it is on some other systems, there is little benefit in reserving SGA expansion space with the `SGA_MAX_SIZE` parameter. It defaults to the actual SGA size.

C.3.9 LOG_ARCHIVE_DEST

This parameter can indicate a pubset, such as `LOG_ARCHIVE_DEST=:PUB1:` to store all the archived redo logs on a special media.

D
Troubleshooting

This topic describes problems that you may encounter when using Oracle Database 19c on BS2000, and provides you with information about how to diagnose and overcome such problems.

To solve a problem, identify the type of the problem and locate the relevant information in this topic. Examine each of the listed points to find the cause of the problem. Carry out the suggested solution, and try again. The event log file described in this document may help you to diagnose the problem.

This appendix contains the following topics:

- Problems Creating an Oracle Database
- Problems Starting an Oracle Database Instance
- Problems When Starting the Background Tasks
- Problems Accessing Database and Log Files
- Oracle Database Trace Files

> **See Also:**
>
> Oracle Error Messages for BS2000 in this guide and *Oracle Database Error Messages Reference* for information about specific messages

D.1 Problems Creating an Oracle Database

You should always use the BS2000 procedure `INSTALL.P.SUPER` to create a new database, because this is the easiest way to get a correct instance. If you encounter problems during this process, then study the diagnostic output, correct, and run the respective part manually, or remove the partially created database, and rerun the whole process.

Also, check the following:

- If the BS2000 user ID has sufficient `PUBLIC-SPACE-LIMIT` for the corresponding pubset.
- If enough disk space is available on the pubset that is used to create the databases.
- If the disk space fragmentation is too high.

D.2 Problems Starting an Oracle Database Instance

This topic contains information related to problems that you may encounter when starting an Oracle Database instance.

- If you get an ORA-05032 error with no extra information, then check the following:

  If you attempt to start an instance and the startup fails, you might get an ORA-05032 message and not much information. This indicates that a problem occurred in a very early stage of the startup, when Oracle Database error stack and backtracking mechanism were not yet active. If this is the case, then you should check the following:

  - If you called the ORAENV procedure prior to calling SQL*Plus.
  - If you specified a correct and unique ORASID value in the ORAENV file
  - If there potential address range conflicts:

    The address ranges assigned to the kernel memory pool, the SGA, and the PGA, in each task, could be partially occupied by shared BS2000 subsystems also used in the instance. Contact the System Administrator to find out how the subsystems are arranged. Then change the corresponding xxx_BASE environment variables in the ORAENV file to relocate the Oracle Database areas to suitable address ranges.
  - If the user address space is large enough:

    A small address space limit may not leave enough space for Oracle Database requirements.
  - If a previous startup attempt failed, leaving invalid background, server, or user tasks:

    If the Oracle Database has not been shut down properly, then the old background tasks or server tasks may hang and may still be connected to the SGA of the old instance. This inhibits the creation of a new SGA. You may get a message indicating shutdown in progress.

    Cancel the remaining background, server, and user tasks. Exit SQL*Plus, which is required to release shared memory pools of the old instance and retry.

D.3 Problems When Starting the Background Tasks

If you get a timeout message when starting the background tasks, then check the following points:

- If the background tasks are blocked in the BS2000 job queue. This may occur due to a system overload or an insufficient task priority.

  The background tasks must always be started with the IMMEDIATE option and preferably in a reserved BS2000 jobclass. Check the BGJPAR environment variable and the user attributes of the BS2000 user ID. Cancel any background tasks that have already started.
- If no background task can be found using the /SHOW-USER-STATUS command, then the jobs have probably been aborted. Check the job outputs.

D.4 Problems Accessing Database and Log Files

If you have problems opening, closing, reading, or writing a database or log file, then check the following points:

- If the file exists.

- If the file is accessible to the program that is trying to open it.
- If there is a hardware problem.
- If you specified the correct block size.

If you specified the `ORAENV` environment variable, `SF_PBLKSIZE`, at database creation, then you must continue to use the same specification whenever you run an `ALTER DATABASE` statement.

D.5 Oracle Database Trace Files

Whenever an Oracle database encounters an exception, it writes a trace or a dump file. Verify this trace file for more detailed information about the issue. You may need to send the trace file to the Oracle Support Services Representative, if you cannot resolve the issue.

E

File Types and Names Used by Oracle

The following is a list of file types and names used by Oracle Database on BS2000:

sid.DBS.xxx.DBF

Database files such as *sid*.DBS.CONTROL.DBF, *sid*.DBS.DATABASE1.DBF, or *sid*.DBS.LOG1.DBF contain the entire Oracle database including data dictionary, user tables, log files, and so on.

sid.DBS.INIT.ORA

Parameter file used when an instance is started.

sid.P.ORAENV

Environment definition file, which defines the program environment. You must run this file before starting an instance or an application:

/CALL-PROCEDURE *sid*.P.ORAENV

S.E.tsn.YYYY-MM-DD.hh.mm.ss

Temporary ENTER-PROCEDURE file for starting the background processes.

S.OUT.tsn.YYYY-MM-DD.hhmmss.number

Temporary files of background processes containing the runtime information. After a task is successfully completed, these files are removed. These files can help in diagnosis.

L.sid.xxxx.SYSOUT.tsn

Run-time listing documenting the start of a background process. When a process finishes, orderly listings are removed, except the SDF-P variable BGJOUT is set to KEEP in *sid*.P.ORAENV for diagnosis purposes. *xxxx* corresponds with the identifier of Oracle background jobs.

Oracle Net Services LOG files

All errors encountered in Oracle network products are appended to a log file.

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for more information about Oracle Net Services log files

T.sid.INSTALL.E.SUPER

Temporary file created by the BS2000 procedure `INSTALL.P.SUPER`. You can remove this file after a successful Oracle Database installation.

T.sid.INSTALL.E.SUPER.72

Temporary file created by the BS2000 procedure `INSTALL.P.SUPER`. You can remove this file after a successful Oracle Database installation.

L.sid.xxx.LOG

Log file created by the BS2000 procedure `INSTALL.P.SUPER`, for example, `L.`*`sid`*`.CATALOG.LOG`. You can remove this file after a successful Oracle Database installation.

ORAXALOG.tsn-NULL-YYMMDD.TRC

This file is used to trace errors when openUTM is used with Oracle Database. This file is created only when an error occurs or when tracing is turned on. You can replace `NULL` by a `db_name` when specified in the open string.