Oracle® Database

User's Guide

11*g* Release 2 (11.2) for Fujitsu BS2000/OSD

E27507-02

October 2012

ORACLE®

Oracle Database User's Guide, 11*g* Release 2 (11.2) for Fujitsu BS2000/OSD

E27507-02

Primary Author: Tanvee Ravi

Contributing Author: Janelle Simmons

Contents

Index

Preface

This manual, with the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*, forms the system-specific component of a set of manuals which document installation, maintenance, and use of the Oracle Database 11*g* Release 2 (11.2).

General information about the Oracle Database for all operating systems is contained in the Oracle Database documentation set.

This Preface contains the following topics:

Audience

This manual is intended for users of the Oracle Database running under the BS2000/OSD operating system, and for those who provide support to these users. Those responsible for installing the Oracle Database, or administering the Oracle Database, or both, should also refer to the information contained in the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

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 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

The Oracle Database products that run under BS2000/OSD are identical, in the way in which they are supported, to the Oracle Database products that run under any other operating system. 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 under 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 11*g* Release 2 (11.2) for BS2000/OSD, providing them with BS2000/OSD-specific information about using Oracle Database products. It does not describe how to use a product unless its use is different than that described in the generic documentation. System programmers and database administrators responsible for installing the Oracle Database, or administering the Oracle Database, or both, should read this manual as well as the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*. There are places where the information in these manuals overlap and is presented differently depending on the target audience.

The reader is assumed to have a fundamental knowledge of BS2000/OSD. No attempt is made to document features of BS2000/OSD except as they affect or are affected by the Oracle Database 11*g* Release 2 (11.2).

Related Documents

For more information, refer to the following resources:

- Fujitsu documentation at `http://manuals.ts.fujitsu.com/index.php?cookie`
- *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*

Printed documentation is available for sale in the Oracle Store at

`http://shop.oracle.com`

To download free release notes, installation documentation, white papers, or other collateral, visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technetwork/community/join/overview/index.html`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technetwork/indexes/documentation/index.html`

Conventions Used in this Manual

The following conventions are observed 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.
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 which 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	#
parantheses	()
period	.
semicolon	;

Name	Symbol
single quotation mark	'

1 Getting Started

This chapter provides the BS2000/OSD-specific information that you must use with Oracle Database 11*g* Release 2 (11.2) for Fujitsu BS2000/OSD.

This chapter contains the following sections:

1.1 The Oracle Database Environment-Definition File

Every Oracle Database utility and product under BS2000/OSD uses an Oracle Database environment-definition file, which is referred to as `ORAENV`. You must generate this file before you use the Oracle Database products as it contains a number of Oracle Database environment variables. These Oracle Database environment variables describe the operating environment for the Oracle Database and utilities. The section "Generating the Environment-Definition File" explains how to create the file.

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 first generating the `ORAENV` file, then you will not be able to connect to the Oracle Database.

The `ORAENV` file is a text file that has the format of a BS2000 command procedure. The command procedure calls itself using the `/SET-FILE-LINK ORAENV,`*`filename`* command. Each line contains an Oracle Database environment variable and its assigned value. When reading this file, Oracle Database ignores all lines that have a slash symbol (/) or asterisk symbol (*) in column one.

1.1.1 Generating the Environment-Definition File

You must generate an `ORAENV` file prior to using Oracle Database for the first time. Perform the following steps to generate an `ORAENV` file:

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

```
/CALL-PROCEDURE $ORAC1120.INSTALL.P.USER
```

where `$ORAC1120` is the name of Oracle Database 11*g* Release 2 (11.2) installation user ID.

You are prompted for the database system identifier, known as the *SID*.

2. Enter the *SID*. If you do not know what the *SID* is, then consult the database administrator.

1.1.2 Calling the Environment-Definition File

Call the ORAENV file (*sid*.P.ORAENV) by entering the CALL-PROCEDURE command on the ORAENV file. For example, to call the example ORAENV file for the database DEMO, enter the following command:

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

1.1.3 Specifying the Environment Variables

The following is the content of the ORAENV file available with Oracle Database:

```
/SET-PROC-OPT DATA-ESCAPE=*STD
/DECL-PAR (SYSCMD(INI-VAL='DEMO.P.ORAENV'))
/       REMARK * SYSCMD must be name of this file
/       WRITE-TEXT '                                          '
/       WRITE-TEXT ' +---------------------------------------+ '
/       WRITE-TEXT ' I  Oracle Database 11g Release 2 (11.2)  I '
/       WRITE-TEXT ' I           environment setup            I '
/       WRITE-TEXT ' +---------------------------------------+ '
/       WRITE-TEXT '                                          '
/       SET-FILE-LINK ORAENV,&SYSCMD
/       SET-FILE-LINK ORALOAD,$ORAC1120.ORALOAD.LIB
/       SET-FILE-LINK ORAMESG,$ORAC1120.ORAMESG.LIB
/&*     MOD-SDF $ORAC1120.SYSSDF.ORACLE.USER
/&*     *** if SYSOUT protocol is desired set BGJOUT='KEEP' ***
/&*     SET-VAR BGJOUT='DEL'
/       EXIT-PROCEDURE
*
*  parameters for users:
*
ORAUID=/BS2/$ORAC1120
ORASID=DEMO
NLS_LANG=German_Germany.D8BS2000
* PRINTPAR=
/END-PROCEDURE
```

If you want, you can change Oracle Database 11*g* Release 2 (11.2) working environment by editing the user variables in this file. The Appendix B, "Oracle Environment Variables" contains a list of the variables you can specify in the ORAENV file. The values you assign to user variables are specific to your task only. The database administrator can also set other variables that affect the whole database instance. If you try to set values for the DBA-specific variables in the ORAENV file, then they are ignored.

Note: You can create an individual ORAENV file for each database with which you work. To set the environment variables, call the ORAENV file containing the environment variables for the database you want to use.

1.2 The ORALOAD Library

The ORALOAD library ($ORAC1120.ORALOAD.LIB by default) is required for starting Oracle Database 11*g* Release 2 (11.2) programs. Oracle Database uses this library to dynamically load executables when required. The ORALOAD library must be identified by the link name ORALOAD prior to calling Oracle Database programs. You get a BLS (BS2000 loader) error message, if the link name is missing. The ORALOAD link name is set when the ORAENV procedure is called. Another library, the ORAMESG library ($ORAC1120.ORAMESG.LIB by default) is required for Oracle messages. This library is assigned the link name ORAMESG in the ORAENV procedure.

1.3 Starting Oracle Utilities with BS2000 SDF commands

Before you start Oracle Database products, you must call the environment-definition file, as described in Section 1.1.2, "Calling the Environment-Definition File".

Start the Oracle Database programs and utilities by entering a START-PROGRAM command with the program name at the BS2000 command prompt (/). Specify the options and operands as the first data-input line when the data prompt (*) is displayed, as shown in the following example:

```
/START-PROGRAM $ORAC1120.program_name
CCM0001 enter options:
* [option_switch] [arguments]
```

where:

program_name is the name of the program or utility to be started

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 ID and password combination, or both.

Example 1–1

To start SQL*Plus, enter the following command:

```
/START-PROGRAM $ORAC1120.SQLPLUS
* /NOLOG
SQL> CONNECT SYS / AS SYSDBA
Enter password: password
```

As soon as the program is loaded, the CCM0001 prompt is displayed to let you enter the command line options. As shown in the preceding examples, you can enter the *option_switch* or *arguments* for the program. The prompt of the program is then displayed, which in the case of SQL*Plus, is SQL>. You can now enter one of the commands of the program. Refer to the generic documentation for the product for a description of the valid commands.

Alternatively you can start the Oracle Database utilities with the following BS2000 SDF command:

```
/START-EXECUTABLE ($ORAC1120.ORALOAD.LIB,program_name)
```

For example, if you want to start SQL*Plus, then enter the command:

```
/START-EXECUTABLE ($ORAC1120.ORALOAD.LIB,SQLPLUS)
```

1.4 Starting Oracle Utilities in UNIX-Style

You can also start the utilities with the following commands:

```
/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-MKWALLET          or     /MKWALLET
/START-ORACLE-SQLLOADER         or     /SQLLDR
/START-ORACLE-SQLPLUS           or     /SQLPLUS
/START-ORACLE-TNSPING           or     /TNSPING
/START-ORACLE-RMAN              or     /RMAN
```

Parameters can be specified after the start command (in quotation marks, if the parameters contains equal to (=) or blanks). Before you start the utility, activate the `MOD-SDF` command in the `ORAENV` file and call the `ORAENV` file.

Example:

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

1.5 Starting Oracle Utilities in the POSIX environment

Starting with Oracle Database 11*g* Release 2 on Fujitsu BS2000/OSD, you can run utilities like SQL*Plus not only in the normal BS2000 environment, but also in the POSIX environment.

During the installation of the Oracle Database software the utilities are installed within the POSIX file system in the directory *oracle_home_path*`/bin`. Before starting Oracle utilities in the POSIX shell, you must set the environment variable `ORACLE_HOME` and extend the environment variable `PATH` by the pathname of the Oracle directory *oracle_home_path*`/bin`. For example:

```
$ ORACLE_HOME=/u01/app/orac1120/product/dbhome
$ export ORACLE_HOME
$ PATH=$ORACLE_HOME/bin:$PATH
$ export PATH
```

Alternatively you can execute the profile *oracle_home_path*`/.profile.oracle` that is created during the installation of the Oracle Database software under POSIX. This profile sets and expands the most important variables like `ORACLE_HOME` and `PATH`. Execute the profile as follows:

```
$ . /u01/app/orac1120/product/dbhome/.profile.oracle
```

Set the variable ORACLE_SID to start an Oracle utility for a specific Oracle instance. Additional instance-specific parameters that are defined in the related BS2000 ORAENV file, may be set in the POSIX environment or by accessing the BS2000 ORAENV file.

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 named oraenvsid in the *oracle_home_path*/dbs directory. This file contains the full qualified BS2000 filename of the BS2000 ORAENV file. It acts like a link to the ORAENV file in the BS2000 file system.

For example, if you want to access the ORAENV file $ORADATA.ORCL.P.ORAENV, then you must create a oraenvORCL file in the *oracle_home_path*/dbs directory as follows:

```
$ ORACLE_HOME=/u01/app/orac1120/product/dbhome
$ 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 subordinated variables.Environment variables in the POSIX shell take precedence over settings in the BS2000 ORAENV file.
- The *SID* in the filename oraenv*sid* is case sensitive and must match the *SID* specified in ORACLE_SID.
- You must grant access for all users to 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 will get the job parameters for the start of a dedicated server in the BS2000 environment from the BGJPAR variable. If this variable is not specified, then Oracle Net Services will use default values.

Note: The BGJPAR variable is not set after running the *oracle_home_path*/.profile.oracle profile.

While using the BEQ protocol, it is recommended that particular BS2000 job parameters are defined for BS2000 jobs 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, if a bequeathed server task should be assigned to a special JOB-CLASS, then 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 the same way as on other UNIX systems, such as for SQL*Plus, using the following commands:

```
$ sqlplus /nolog
```

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

1.6 Connecting to an Oracle Instance

You can connect to an Oracle instance in the following ways:

- Using Oracle Net Services with the Bequeath adapter (Refer to the "Oracle Net Services" chapter in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*)
- Using Oracle Net Services over TCP/IP or IPC

Check with the database administrator to see whether 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 instance as part of the logon string appended to the *userid/password*, and separated from it by an at sign (@), as illustrated in the following sections.

1.6.1 Default Connections

If you do not specify a connection string, then the environment variable `DEFAULT_CONNECTION`, if set, is used to establish the Oracle database connection. Refer to Appendix B, "Oracle Environment Variables" for more information about the `ORAENV` file and the `DEFAULT_CONNECTION` environment variable.

1.6.2 Accessing an Oracle Instance

Access to a local or remote instance is done through Oracle Net Services. Use the Oracle Net Services logon string to identify the following for accessing a local or remote database:

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

The Oracle Net Services logon string has the following structure:

```
/START-PROGRAM $ORAC1120.SQLPLUS
* userid/password@service_name
```

where:

service_name specifies a service name entered 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 here, then contact the database administrator.

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

```
HR/HR@SERVERX
```

For information about connecting to an Oracle Database using the Bequeath adapter, refer to the "Oracle Net Services" chapter in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

1.7 Using BS2000 Files for Input and Output

In most cases, Oracle Database for BS2000/OSD 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.

1.7.1 Text Files

Textual data is usually stored in SAM or ISAM files; each record is taken to be one text line. Examples are 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.

1.7.2 Binary Files

Binary data is usually stored in PAM files.

1.7.3 Generic Oracle Database File Name Syntax

The convention used in generic Oracle Database documentation represents file names as two parts separated by a period, as in LOGIN.SQL. This syntax is correct for BS2000. However, as there is no "current directory" concept in BS2000, you have to add a prefix to the generic example names to get a full BS2000 file name.

1.7.4 Default File Name Extensions

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

	Original File Name	Extended File Name
1.	TEST.TEST	TEST.TEST.EXT
2.	TST	TST.EXT
3.	T.T	T.T
4.	TEST.TST	TEST.TST

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

1.7.5 Using Link Names

Instead of specifying a file name, in special cases, you can also refer to the link name of a previously issued BS2000 /SET-FILE-LINK command by using 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 the link=*linkname* notation cannot be used.

Note: When using the `link=linkname` notation, default file name extensions do not work. As a result, file name defaults derived from such notation are not valid, and you have to 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.

1.7.6 Fixed Link Names

Oracle Database 11*g* Release 2 (11.2) for BS2000/OSD uses fixed link names for specific files.

The most important of these are as follows:

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

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

2

Oracle Database Utilities

This chapter describes how you start the following Oracle Database utilities:

- SQL*Loader
- The Export Utility
- The Import Utility
- The Data Pump Export Utility
- The Data Pump Import Utility
- Oracle Text Loader

For a more detailed discussion of these utilities, refer to *Oracle Database Utilities*.

2.1 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. Refer to the *Oracle Database Utilities* manual for a detailed description of SQL*Loader and its demonstration files.

For restrictions when using SQL*Loader refer to the section, " Known Problems, Restrictions, and Workarounds" in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

2.1.1 Starting the SQL*Loader Utility

To start SQL*Loader, enter the following command:

```
/START-PROGRAM $ORAC1120.SQLLDR
* SCOTT/password
```

2.1.2 Using the SQL*Loader Demonstration Files

The demonstration files are shipped under:

```
$ORAC1120.RDBMS.DEMO.ULCASE*.CTL
$ORAC1120.RDBMS.DEMO.ULCASE*.SQL
$ORAC1120.RDBMS.DEMO.ULCASE*.DAT
```

To run the ULCASE1 demo use the following steps:

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

```
START-PROGRAM $ORAC1120.SQLPLUS
* SCOTT/password
SQL> START $ORAC1120.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-PROGRAM $ORAC1120.SQLLDR
* SCOTT/password $ORAC1120.RDBMS.DEMO.ULCASE1 ULCASE1 ULCASE1
```

2.2 The Export Utility

The Export utility is used to write data from an Oracle Database into the BS2000 system files. Use this utility with the Import utility to back up your data, and to move data between Oracle Databases.

For restrictions when using the Export utilities refer to the section, " Known Problems, Restrictions, and Workarounds" in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

2.2.1 Starting the Export Utility

To start the Export utility, EXP, enter the following command:

```
/START-PROGRAM $ORAC1120.EXP
* SCOTT/password
```

If you omit the SCOTT/*password* parameters, then you are prompted for them.

Export dump files are usually created by EXP as SAM files. You can override default output file specifications by running a file command such as:

```
/FILE expfile,LINK=explink,FCBTYPE=SAM,RECFORM=F, -
BLKSIZE=(STD,1),RECSIZE=2048
```

Then, call EXP 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:

```
/FILE expfile,LINK=explink,FCBTYPE=SAM,RECFORM=F, -
BLKSIZE=(STD,1),RECSIZE=2032
```

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 a large volume of data, the default disk-space allocation for the output file will be inappropriate, and the program will spend a significant amount of time allocating secondary extents of disk space. If the maximum number of extents exceeds the number that the catalog entry can hold, then an output-file error will occur.

As a counter-measure, you should always preallocate the EXP output file with the BS2000 /FILE command, prior to starting the Export utility. When allocating the file, you should use a realistic estimate for both the primary and secondary space allocations.

For example:

```
/FILE LARGE.EXPORT.DMP,LINK=EXPOUT,SPACE=(3000,3000)
/START-PROGRAM $ORAC1120.EXP
* system/manager
...
Export file: EXPDAT.DMP >link=expout
...
```

2.2.2 Exporting to Foreign Systems

You can export to foreign systems using the following methods:

- Transferring Data by Tape
- Transferring Data by File Transfer

2.2.2.1 Transferring Data by Tape

To export directly to tape, enter a FILE command as follows:

```
/FILE tapefile,LINK=tapelink,FCBTYPE=SAM,RECFORM=F, -
BLKSIZE=2048,RECSIZE=2048,DEV=<device>,VOL=<vsn>
```

Then, call EXP, specifying the following value in response to the output file name prompt:

```
LINK=tapelink
```

You also must set the EXP_CLIB_FILE_IO environment variable to FALSE in the ORAENV file.

The export utility writes the output as SAM files, which simplifies export to an Oracle Database on foreign systems.

2.2.2.2 Transferring Data by File Transfer

If you use FTP, then ensure that you specify binary mode (to avoid automatic EBCDIC-ASCII conversion).

2.3 The Import Utility

The Import utility is used to write data from the files created by the Export utility to an Oracle Database.

For restrictions when using the Import utility refer to the section, "Known Problems, Restrictions and Workarounds" in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

2.3.1 Starting the Import Utility

To start the Import utility, IMP, enter the following command:

```
/START-PROGRAM $ORAC1120.IMP
* SCOTT/password [options]
```

If you omit the SCOTT/*password* parameters, then you are prompted for them.

2.3.2 Importing from Foreign Systems

This section gives you some guidelines on importing data from non-BS2000 systems.

2.3.2.1 Import File Block Size

If the import file on the BS2000/OSD 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.

2.3.2.2 Transferring Data by 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, enter a FILE command such as the following:

```
/FILE tapefile,LINK=tapelink,DEV=T-C1,STATE=FOREIGN
```

Then, call IMP, specifying the following as the input file name at the input file name prompt:

```
LINK=tapelink
```

You must also set the IMP_CLIB_FILE_IO environment variable to FALSE in the ORAENV file.

2.3.2.3 Transferring Data by File Transfer

If you use FTP, then ensure that you specify binary mode (to avoid automatic ASCII-EBCDIC conversion). The received file will be stored as a PAM file by the BS2000 FTP utility and can immediately be used as an input file to IMP.

When you try to use an export file from BS2000 as an import file on an ASCII platform use as transfer utility FTP on BS2000 side and indicate the parameters **binary** and **ftyp binary** in order to avoid insertion of NL (new lines) at block boundaries.

2.4 The Data Pump Export Utility

Data Pump Export and Import are functionally similar to Export and Import discussed previously, but all of the I/O processing for dump files is done in the Oracle Database server rather than in the client utility session.

The Data Pump Export utility is used to write data from an Oracle Database into the BS2000 system files. Use this utility with the Data Pump Import utility to backup your data, and to move data between Oracle databases.

2.4.1 Starting the Data Pump Export Utility

To start the Data Pump Export utility, EXPDP, enter the following command:

```
/START-PROGRAM $ORAC1120.EXPDP
* username/password [options]
```

Data Pump Export dump files are created by `EXPDP` as `PAM` files with `BLKSIZE=(STD,2)`.

When you try to use an export file from BS2000 as an import file on an ASCII platform, use as transfer utility FTP on BS2000 side and indicate the parameter `binary`.

Note: If you start `EXPDP` 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 Export to tape is not supported.

2.5 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.

2.5.1 Starting the Data Pump Import Utility

To start the Data Pump Import utility, `IMPDP`, enter the following command:

```
/START-PROGRAM $ORAC1120.IMPDP
* username/password [options]
```

If you use an export file from an ASCII platform as an import file on BS2000, then use as transfer utility FTP on BS2000 side and indicate the parameter `binary`.

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 from tape is not supported.

2.6 Oracle Text Loader

This utility imports and exports text data. For installation of Oracle Text, refer to the chapter, "Oracle Text" in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

To start Oracle Text Loader enter the following command:

```
/START-PROGRAM $ORAC1120.CTXLDR
*-USER username/password [options]
```

For more information, refer to *Oracle Text Reference*.

3

SQL*Plus

SQL*Plus is a general purpose utility used for running SQL commands and PL/SQL blocks, perform database administration, and so on. This chapter describes how you use SQL*Plus under BS2000/OSD. It supplements the *SQL*Plus User's Guide and Reference* with information about the following topics:

- Running SQL*Plus
- Specifying the Search Path for SQL*Plus Command Files
- Sample Schemas and SQL*Plus
- SQL*Plus Limits
- Using SQL*Plus in the POSIX environment

3.1 Running SQL*Plus

The following sections describe how to run SQL*Plus under BS2000/OSD:

- SQL*Plus User Profiles
- Starting SQL*Plus
- Interrupting SQL*Plus
- Issuing BS2000 Commands from SQL*Plus
- Starting the BS2000 Editor
- The SQL ASCII Function
- Spooling SQL*Plus Output
- Using SQL*Plus Symbols

For restrictions related to using SQL*Plus, refer to the section, Known Problems, Restrictions, and Workarounds in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

3.1.1 SQL*Plus User Profiles

There are two startup files for SQL*Plus:

- `GLOGIN.SQL`, which is the global startup file
- `LOGIN.SQL`, which is designed for local and individual use

Whenever any user starts SQL*Plus, first the `GLOGIN.SQL` file is read, followed by the user's `LOGIN.SQL` file.

3.1.1.1 The GLOGIN.SQL Global Startup File

The global startup file GLOGIN.SQL begins whenever any user starts SQL*Plus. This file can contain SQL statements or SQL*Plus commands to be run at the beginning of the SQL*Plus session. The GLOGIN.SQL file is located under the $ORAC1120 user ID, and its name is $ORAC1120.SQLPLUS.ADMIN.GLOGIN.SQL. The database administrator may customize the GLOGIN.SQL file if required. The GLOGIN.SQL file will be run regardless of the current user ID.

3.1.1.2 The LOGIN.SQL User Startup File

The LOGIN.SQL startup file is called after the GLOGIN.SQL command file each time a user starts SQL*Plus. Like GLOGIN.SQL, this file may contain either SQL statements or SQL*Plus commands that a user wants to run at the beginning of every SQL*Plus session.

SQL*Plus first searches for LOGIN.SQL under the current BS2000 user ID. If the LOGIN.SQL file is not found, but the SQLPATH environment variable specifies a path, then SQL*Plus searches along that path. If SQL*Plus finds a LOGIN.SQL file, then it runs the first LOGIN.SQL file that it finds. For a customized SQL*Plus environment, each user ID can have its own LOGIN.SQL file.

Refer to Appendix B, "Oracle Environment Variables" for a description of the SQLPATH environment variable. Refer to the *SQL*Plus User's Guide and Reference* for more information about LOGIN.SQL.

The following is a sample startup file:

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

3.1.2 Starting SQL*Plus

To start SQL*Plus, enter:

```
/START-PROGRAM $ORAC1120.SQLPLUS
* userid/password
```

If you omit either the user ID or password, then you are prompted to enter them.

Once you are logged in to SQL*Plus, the SQL prompt is displayed:

```
SQL>
```

If you do not want to be prompted for user ID/password, then enter:

```
/START-PROGRAM $ORAC1120.SQLPLUS
* /NOLOG
SQL> connect userid/password
```

You can enter any SQL statement (SELECT... FROM, CREATE TABLE, and so on.) or any SQL*Plus command (SET LINESIZE, COLUMN x FORMAT. . ., and so on) in response to this prompt.

3.1.3 Interrupting SQL*Plus

Use the INTERRUPT key [K2] to interrupt SQL*Plus SQL statements. For example, you can interrupt SQL*Plus if you receive a long report that you do not want to select.

When you press the INTERRUPT key [K2], the Oracle Database stops retrieving rows and returns you to the SQL*Plus command level.

Note: If you issue an INTERRUPT when input is requested, then you must answer this request before the processing can be interrupted. However, this answer will be ignored.

3.1.4 Issuing BS2000 Commands from SQL*Plus

The SQL*Plus `HOST` command and the `$` command enable you to enter a BS2000 command while you are logged on to SQL*Plus.

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

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

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 command level:

  ```
  SQL> HOST
  ```

  To return to SQL*Plus, you must use the `RESUME` command.

- If you enter the `HOST` command with a BS2000 command, then the command is run and you return to SQL*Plus:

  ```
  SQL> HOST STA L
  ```

3.1.5 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`
- starts the editor EDT, which reads the file `SQLEDT.BUF` into the work area

You can then edit and write to this file (using the `@write` command). Using the `@halt` command, you can leave the editor and return to SQL*Plus. SQL*Plus then reads the current contents of `SQLEDT.BUF` back into its command buffer, from which the SQL statement can be run.

Note: If you used the SQL*Plus `DEFINE _EDITOR` command to define a name for the editor, then BS2000 will ignore it. It always starts EDT.

You can also use the EDIT command to edit a SQL file by specifying the SQL file in the EDIT command. For example, if you enter the following command, then the editor EDT is called to edit the LOGIN.SQL file. Note that you can omit the default file name extension .SQL.

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

Refer to the *SQL*Plus User's Guide and Reference* for more details about the SQL EDIT command.

3.1.6 The SQL ASCII Function

The ASCII function takes a character (under BS2000/OSD, this will be an EBCDIC character) and returns the numerical representation of that character in the given character set. The ASCII function does not convert an EBCDIC character into its ASCII equivalent. For example, the ASCII function returns the value 193 for the character A. The inverse function is CHR, for example, CHR(193)='A'.

3.1.7 Spooling SQL*Plus Output

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

> **Note:** The output generated by BS2000/OSD operating system commands will not be spooled.

When you issue a SPOOL OUT request, the program issues 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. If you need to specify any /PRINT command options, such as character sets, or routing to a remote printer, then do so by adding the following line to the ORAENV file:

```
PRINTPAR=options
```

Where options is any sequence of/PRINT command options (refer to the BS2000/OSD manual, *Benutzerkommandos (ISP-Format)* for more information about these options). The program then issues a /PRINT command, which includes these options.

3.1.8 Using SQL*Plus Symbols

The SQL symbol used for negation is the exclamation point (!). The use of exclamation point is recommended when specifying "not equal," especially for applications that may be run in different environments.

If no exclamation point is available on your keyboard, then you can use left and right angle brackets (<>) for "not equal."

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

3.2 Specifying the Search Path for SQL*Plus Command Files

When you use the following command, SQL*Plus searches for a file called filename.SQL under the current BS2000 user ID:

```
SQL> START filename
```

If this file cannot be found, then SQL*Plus searches the paths specified by the ORAENV environment variable SQLPATH. This variable is used to specify one or more file name prefixes separated by a semicolon (;), which should be applied when searching for the command file.

For example, if SQLPATH is set to PRIVATE and $GLOBAL, as follows:

```
SQLPATH=PRIVATE;$GLOBAL
```

then, when you enter the following command:

```
@filename
```

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

1. *filename*.SQL
2. PRIVATE.*filename*.SQL
3. $GLOBAL.*filename*.SQL

Refer to Chapter 1, "Getting Started" for more information about default file name extensions.

3.3 Sample Schemas and SQL*Plus

The sample schemas provide a common platform for examples. For more information about the sample schemas and SQL*Plus, Refer to *SQL*Plus User's Guide and Reference*.

Refer to the chapter "Creating and Upgrading a Database" in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD* for information about how to install the sample schemas.

3.4 SQL*Plus Limits

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

Item	Limit
File name length	54 (including catalog-id and user ID)
LINESIZE	32767
MAXDATA	32767
Maximum number of nested command files	12

3.5 Using SQL*Plus in the POSIX environment

Starting with Oracle Database 11*g* Release 2 on Fujitsu BS2000/OSD 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

3.5.1 Starting SQL*Plus

You can start SQL*Plus in the POSIX shell. Refer to "Starting Oracle Utilities in the POSIX environment" for more information about running SQL*Plus in the POSIX shell.

3.5.2 Running Shell Commands From SQL*Plus

The SQL*Plus `HOST` command enables you to enter a POSIX shell command, while you are logged on to SQL*Plus.

Keep the following points in mind when using the HOST command:

- If you enter the `HOST` command without any shell command, then it takes you to the command level. To return to SQL*Plus, you must use the `exit` command in the POSIX subshell.
- If you enter the `HOST` command with a shell command, then the command is executed and you return to SQL*Plus.
- Use the `bs2cmd` POSIX shell command to execute BS2000 SDF commands.

3.5.3 Using an Editor in SQL*plus

Start a text editor in SQL*Plus with the `EDIT` command, if you want 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 the default editor in SQL*Plus is set to `edtu`. If the POSIX shell is started by a remote X-client through `rlogin` or `ssh` using a xterm terminal, then the default editor in SQL*Plus is set to `vi`.

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, if you want to define the editor, used by the `EDIT` command, to be the POSIX editor `edtu`, then 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.

4

PL/SQL

PL/SQL is an extension to the SQL language and is used to create, store, modify, retrieve, and manage information in an Oracle database.

This chapter supplements the *Oracle Database PL/SQL Language Reference*, with operating system-specific information about the PL/SQL demonstrations.

See your database administrator if the PL/SQL demonstrations have not already been loaded.

4.1 PL/SQL Demonstrations

The PL/SQL demo scripts do not specify fully qualified file names when including other scripts. To include the necessary prefixes for these files, use the `ORAENV` environment variable, `SQLPATH` (search path for SQL files), as shown in the following example:

```
SQLPATH=$ORAC1120.PLSQL.DEMO;<other prefixes>
```

> **Note:** Before you run PL/SQL, ensure that your database administrator has run the necessary initialization scripts. Refer to the chapter "Creating and Upgrading a Database " in the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

4.1.1 PL/SQL Demos for the Kernel

The demos for the kernel are as follows:

```
PLSQL.DEMO.PLS-EXAMP1.SQL
PLSQL.DEMO.PLS-EXAMP2.SQL
PLSQL.DEMO.PLS-EXAMP3.SQL
PLSQL.DEMO.PLS-EXAMP4.SQL
PLSQL.DEMO.PLS-EXAMP5.SQL
PLSQL.DEMO.PLS-EXAMP6.SQL
PLSQL.DEMO.PLS-EXAMP7.SQL
PLSQL.DEMO.PLS-EXAMP8.SQL
PLSQL.DEMO.PLS-EXAMP11.SQL
PLSQL.DEMO.PLS-EXAMP12.SQL
PLSQL.DEMO.PLS-EXAMP13.SQL
PLSQL.DEMO.PLS-EXAMP14.SQL
PLSQL.DEMO.PLS-SAMPLE1.SQL
PLSQL.DEMO.PLS-SAMPLE2.SQL
PLSQL.DEMO.PLS-SAMPLE3.SQL
PLSQL.DEMO.PLS-SAMPLE4.SQL
```

To run these demos, you must first build the demo tables with PLSQL.DEMO.PLS-EXAMPBLD.SQL and load them with PLSQL.DEMO.PLS-EXAMPLOD.SQL (under any user ID). The following example shows how to do this and run PLS-SAMPLE1:

```
/START-PROGRAM $ORAC1120.SQLPLUS
* user/password
SQL> @PLS-EXAMPBLD
SQL> @PLS-EXAMPLOD
SQL> @PLS-SAMPLE1
```

4.1.2 PL/SQL Demos for Precompilers

The demos for the precompilers are as follows:

```
PLSQL.DEMO.PLS-EXAMP9.PC
PLSQL.DEMO.PLS-EXAMP10.PC
PLSQL.DEMO.PLS-SAMPLE5.PC
PLSQL.DEMO.PLS-SAMPLE6.PC
```

Before running these demos, you must ensure that these demos are compiled and linked as described in Chapter 5, "Programmatic Interfaces".

Note: You must run the RDBMS demos before loading the precompiler demos.

5

Programmatic Interfaces

This chapter provides BS2000/OSD-specific information that supplements the documentation for the individual precompilers (such as Pro*C), and host language calls (Oracle Call Interface). It includes information about the following topics:

5.1 Overview

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, which allows high-level language applications to access data in an Oracle Database by making direct calls to the Oracle Database kernel

Under BS2000/OSD, the Oracle Database precompilers support programs written in C, C++, and COBOL programming languages.

For more detailed information about Oracle Precompilers, refer to *Oracle Database Programmer's Guide to the Oracle Precompilers* and the appropriate supplementary publications from the following list:

- the *Pro*C/C++ Programmer's Guide*
- the *Pro*COBOL Programmer's Guide*

5.1.1 Architecture of the Programmatic Interfaces

All precompiler and Oracle Call Interface (OCI) applications are link-edited with a small stub module. The stub module dynamically loads the bulk code of the Oracle precompiler software from the `ORALOAD` library (by using the `BIND` system macro). 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 together; any attempt to do so results in execution errors. The entries into the 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, whereas for OCI C, the numeric arguments are by value).

Oracle precompilers generate different `SQLLIB` function names for different languages. The following names are used:

- `SQ0XXX`: COBOL
- `SQ2XXX`: C

5.1.2 PL/SQL

The precompilers support PL/SQL as described in the *Oracle Database PL/SQL Language Reference*. 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` you must also specify `USERID=username/password`.

5.2 Building and Running a Programmatic Interface Application

Perform the following steps to build and run a programmatic interface application. For additional details, refer to the specific notes for the programmatic interfaces in this chapter.

1. Edit your source code, including embedded SQL, as outlined in the generic precompiler documentation.
2. Pre-process the source with the corresponding pre-processor.

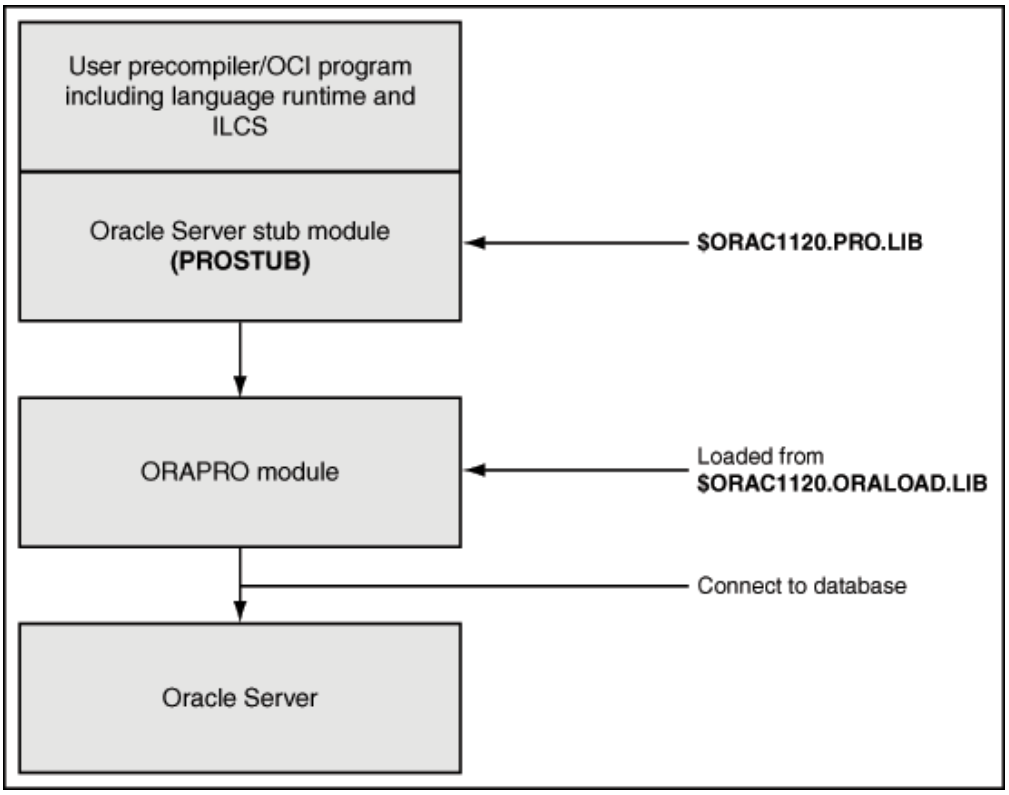
> **Note:** You must use WE8BS2000 as client character set during precompilation (set in `ORAENV` file). Any other character set might lead to problems with concatenation sign ("||").
>
> You do not need to precompile if you are building an OCI C or an OCI COBOL application.

3. Compile the application.
4. Link-edit the application, including the stub module `PROSTUB` from the `PRO.LIB`.
5. Identify the `ORALOAD` library by using a `SET-FILE-LINK` command. (Usually, this is included in the `ORAENV` procedure).
6. Run the application with the `START-PROG` command. The supporting Oracle Database user module is dynamically loaded from the `ORALOAD` library.

7. You can find sample files in the installation user ID: `$ORAC1120.P.PROC`, `$ORAC1120.P.PROCOB` and `$ORAC1120.P.PROLNK`.

Figure 5–1 illustrates the sequence of events outlined in the preceding numbered list and how the programmatic interfaces make use of the program libraries.

Figure 5–1 Usage of Program Libraries by Programmatic Interfaces



5.2.1 Existing Applications

Existing Applications must be pre-processed, compiled and linked anew.

5.2.2 Precompilers

Oracle Database precompilers on BS2000/OSD support LMS libraries for the following files:

- Include Files
- User-Specific Configuration Files
- Input, Output, and List-files

This functionality improves the possibility of saving disk resources and provides you clarity by grouping files in different libraries.

All LMS-library elements to be used must be of element-type "S." Pro* generates elements of type "S" if libraries are used. When you use LMS elements, the precompiler builds temporary files with the prefix "#T.", which are deleted when the preprocessing completes successfully.

When you use LMS library elements, the element name you specify must be the full element name including the suffix. Pro* does not append the suffix to the element name.

5.2.2.1 Include Files

All standard include files are shipped in the LMS library, `$ORAC1120.PRO.INCLUDE.LIB`. You must enter either this library or a user-defined include library for `EXEC SQL INCLUDE` statements using the `INCLUDE` precompiler option as follows:

```
* INCLUDE=$ORAC1120.PRO.INCLUDE.LIB \
* INCLUDE=mylibrary
```

where *`mylibrary`* is the BS2000 file name of the user-defined library, such as `PROC.INCLIB`.

Attention: The order in which you specify different INCLUDE-options affects the performance of precompilation. You should place commonly-used files before rarely-used ones.

5.2.2.2 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.

Note: You must use brackets when specifying the configuration element, as shown in the following example.

For example:

```
* CONFIG=CONFIG.LIB[PROCOB.CFG]
```

5.2.2.3 Input, Output, and List-files

Besides using BS2000 files, you can also benefit from using LMS-elements for precompiler I/O using the options `INAME`, `ONAME`, and `LNAME`.

If you do not specify a library file name and an element from it, then the Oracle precompilers generate ISAM BS2000 files by default. The only option that you *must* enter is the `INAME` option. That can be either a BS2000 file name (`SAM` or `ISAM`) or a library file name and the name of an element from it.

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 appropriate element.

In the following example, Pro*C generates a BS2000-ISAM-output file called SAMPLE.C as the ONAME option has been omitted:

```
* INAME=INPUT.LIB[SAMPLE.PC] \
* LNAME=LIST.LIB[SAMPLE.LST]
```

5.2.3 Additional Remarks

The following are additional remarks on this release of Oracle Database 11*g* Release 2 (11.2) for Fujitsu BS2000/OSD.

- 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 Standard Linkage conventions, then it terminates the task and displays 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.

5.3 Pro*C/C++

This section discusses the procedure for using Pro*C/C++.

5.3.1 Starting Pro*C

To start the Pro*C precompiler, enter the following:

```
/START-PROGRAM $ORAC1120.PROC
* INAME=myprog.PC ONAME=myprog.C [options]
```

where:

myprog is the name of the C program you want to compile and link.

options is one of the PROC options. Refer to the *Pro*C/C++ Programmer's Guide* for a list and description of the valid options.

Note: Unlike as described in *Pro*C/C++ Programmer's Guide*, you must use one Precompiler-option INCLUDE for each path you want to specify. A list as allowed for the option SYS_INCLUDE may cause the precompiler to loop. See Include-Option for Pro*C/C++.

5.3.2 Pro*C Include, System Configuration and Demo Files

The Pro*C include files, demo files, and system configuration file are shipped under:

```
$ORAC1120.PRO.INCLUDE.LIB
```

```
$ORAC1120.C.DEMO.*.PC
$ORAC1120.UTM.DEMO.*.PC
$ORAC1120.CONFIG.PCSCFG.CFG
```

An example of a compilation and precompilation procedure is included in the Oracle Database Software under the name $ORAC1120.P.PROC.

5.3.3 SQLLIB Calls

If you want to code explicit C calls to SQLLIB functions, then you must call SQ2*XXX* instead of SQL*XXX*. For example, call SQ2CEX instead of SQLCEX.

5.3.4 Linking Pro*C

To link a Pro*C program, you need:

- The Common Run-Time Environment, CRTE.
- The Pro* library ($ORAC1120.PRO.LIB), which contains the stub module, PROSTUB. At run time, this module loads the pre-linked module, ORAPRO, which contains the actual SQLLIB code.

> **Note:** You must use BINDER instead of TSOSLNK.

To link your program, you should create your user-specific link procedure. An example of such a link procedure is included in the Oracle Database Software under the name, $ORAC1120.P.PROLNK.

5.3.5 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 in the C programs to verify that you have called the functions correctly.

In the Pro*C programs add the following line:

```
EXEC SQL INCLUDE SQLCPR
```

as you would for SQLCA or SQLDA.

5.3.6 UTM Applications

You can use Pro*C to write UTM program units. Refer to *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD* for UTM programming rules.

5.4 Pro*COBOL

This section discusses the procedure for using Pro*COBOL. You must follow these special considerations, when using Pro*COBOL:

- Host variables of the type PIC S9(*n*) with *n* =8 and *n >10* are not supported.
- When using Pro*COBOL be careful about the following constructions 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 `SQL-LABEL1` is reached. The workaround for this problem is to use `SECTIONS` or to run `PERFORM COB-LABEL1 THRU COB-LABEL2`.

A `COPY` statement as first statement in `WORKING STORAGE SECTION` may result in wrong code generation if copied structures are to be continued by non-copied code because the precompiler generates its data definitions before the first data definition of the source program. To avoid this action, insert one `FILLER` definition as first line in `WORKING-STORAGE SECTION` as follows:

```
01 FILLER PIC X
```

- The default data type for `PIC X` variables has changed in version 8.1.7 from `VARCHAR2` to `CHARF`. A precompiler option provides backward compatibility: `PICX={VARCHAR2 | CHARF (default)}`.

This option is allowed only on the command line or in a configuration file. The new default action is consistent with the usual COBOL move action.

> **Note:** Using the default `PICX=CHARF` while precompiling existing applications may result in run-time error `ORA-1403: no data found`.

For more details, refer to the chapter "Precompiler Options", in *Pro*COBOL Programmer's Guide*.

5.4.1 Starting Pro*COBOL

To start the Pro*COBOL precompiler, enter the following command:

```
/START-PROGRAM $ORAC1120.PROCOB
* INAME=myprog.PCO ONAME=myprog.COB [options]
```

where:

myprog specifies the COBOL program to compile and link

options is one of the PROCOB options described in the *Pro*COBOL Programmer's Guide*.

Note: The *PROCOB* option MAXLITERAL defaults to 180, not 256, as shown in the *Pro*COBOL Programmer's Guide*. The option FORMAT=TERMINAL is not supported.

5.4.2 Pro*COBOL Include, System Configuration, and Demo Files

The Pro*COBOL include files, demo files, and system configuration file are shipped under:

```
$ORAC1120.PRO.INCLUDE.LIB
$ORAC1120.COBOL.DEMO.*.PCO
$ORAC1120.UTM.DEMO.*.PCO
$ORAC1120.CONFIG.PCBCFG.CFG
```

An example of a compilation and precompilation procedure is included in your Oracle Database Software under the name `$ORAC1120.P.PROCOB2000`.

5.4.3 SQLLIB Calls

If you want to code explicit COBOL calls to `SQLLIB` functions, then call `SQ0`*`XXX`* instead of `SQL`*`XXX`*. For example, call `SQ0ADR` instead of `SQLADR`.

5.4.4 Linking Pro*COBOL

To link a Pro*COBOL program, you need:

- The Common Run-Time Environment, `CRTE`.
- The Pro* Library (`$ORAC1120.PRO.LIB`), which contains the stub module, `PROSTUB`. At run time, this module loads the pre-linked module, `ORAPRO`, which contains the actual `SQLLIB` code.
- Unicode is only supported with COBOL2000. This might generate calls to the BS2000-Macro `NLSCNV`. To resolve the `GNLCNV` entry, use the system XHCS library. Refer to *Fujitsu User's Guide XHCS for BS2000* for more information about the `GNLCNV` entry.

Note: You must use `BINDER` instead of `TSOSLNK`.

To link your program, you should create your own user-specific link procedure. An example of such a link procedure is included on your Oracle Database Software under the name, `$ORAC1120.P.PROLNK`.

5.4.5 openUTM Applications

You can use Pro*COBOL to write openUTM program units. Refer to *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD* for openUTM programming rules. Program units written in Pro*C and Pro*COBOL can be combined.

5.5 The Oracle Call Interface

Under BS2000/OSD, the Oracle Call Interface supports the C and COBOL languages.

When you use the set of host language calls that make up the Oracle Call Interface, you can access the data in an Oracle Database by programs written in the C and

COBOL programming languages. *OCI* calls are fully described in the *Oracle Call Interface Programmer's Guide*.

For restrictions refer to "Known Problems, Restrictions and Workarounds" in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

> **Note:** The precompiler products from Oracle offer a higher level interface to the Oracle Database. One precompiler call is translated to several OCI calls. As the precompilers are simpler to use, and in a few cases offer more or different functionality than OCI, you may prefer to use the precompilers for some applications.

5.5.1 Linking OCI

To link OCI program files, you need:

- The Common Run-Time Environment, *CRTE*.
- The Pro* Library (`$ORAC1120.PRO.LIB`), which contains the stub modules `OCI$COB` and `PROSTUB`. At run time, `PROSTUB` loads the prelinked module `ORAPRO`, which contains the actual `SQLLIB` code.

When linking OCI COBOL programs, `OCI$COB` must always be included **before** `PROSTUB`.

> **Note:** You must use BINDER instead of TSOSLNK.

To link your program, you should create your own user-specific link procedure. An example of such a link procedure is included in your Oracle Database Software under the name `$ORAC1120.P.PROLNK`.

For example, to link your program using this sample procedure, enter the following:

```
/CALL-PROCEDURE $ORAC1120.P.PROLNK,dir,module,TYPE=OCIC
```

or:

```
/CALL-PROCEDURE $ORAC1120.P.PROLNK,dir,module,TYPE=OCICOB
```

where the module to be linked is stored in *dir*`.LIB`.

Example files are shipped under:

```
$ORAC1120.RDBMS.DEMO.*.C
$ORAC1120.RDBMS.DEMO.*.COB
```

5.5.2 Optional Parameters

C does not omit optional parameters. Hence, all parameters must be specified. (Refer to the sample C program).

If a length parameter is -1, then the length is determined by scanning the associated string parameter for a null byte. Missing address parameters may be specified as `NULL`. In C, the -1 should be cast to the proper type.

For COBOL, you may omit optional trailing parameters; the call interface provides default values.

5.6 The Object Type Translator

This section discusses port-specific notes for using the Object Type Translator.

5.6.1 Starting Ott

As the Object Type Translator is based on Java it can only be started in the POSIX environment. You must use the JDBC Thin driver to connect to the database. The connect string is published 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=scott/tiger 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.

5.6.2 Ott System Configuration File

The OTT system configuration file is shipped under:

```
$ORACLE_HOME/precomp/admin/ottcfg.cfg
```

6

Using the Oracle Database Under openUTM

This chapter describes the Oracle Database-specific points that you should consider when using Oracle Database 11*g* Release 2 (11.2) for BS2000/OSD under openUTM (Universal Transaction Monitor). The following topics are discussed in this chapter:

6.1 SQL Operations

UTM application program units must use embedded SQL. Calls to the Oracle Call Interface (OCI) are not allowed.

6.1.1 CONNECT

A connection is implicitly established when the UTM task is started. This connection uses the data specified in the open string. Further explicit `CONNECT` operations issued by the program units are not allowed.

6.1.2 COMMIT

An explicit `COMMIT` statement is not allowed in UTM program units. The openUTM will automatically issue one on a `PEND RE`, `FI`, `SP`, or `FC` operation.

6.1.3 ROLLBACK

An explicit `ROLLBACK` statement is not allowed in UTM program units. The openUTM will automatically issue a `ROLLBACK` statement on encountering a `PEND ER`, `RS`, `FR`, or `RSET` operation.

6.1.4 SAVEPOINT

The `SAVEPOINT` statement is not allowed in UTM program units.

6.1.5 Cursor Operations

A cursor is valid only until a `PEND` is run. Because of a possible task change during a `PEND KP`, `PA`, or `PR`, you cannot perform operations on a previously filled cursor such as `OPEN` or `FETCH` after a `PEND KP`, `PA`, or `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 (Refer to the openUTM manual, *Programming Applications with KDCS for COBOL, C and C++*),

which will wait until input comes from the terminal, or to assign the same TACCLASS to subsequent programs after a PEND PA or PR.

6.1.6 Dynamic SQL

You may use dynamic SQL as described in *Oracle Database Programmer's Guide to the Oracle Precompilers*.

6.1.7 PL/SQL

COMMIT, ROLLBACK, CONNECT, and SAVEPOINT statements are not allowed in PL/SQL programs running under UTM.

6.1.8 Autocommit

Autocommit operations should be avoided because they violate the synchronization between Oracle Database and UTM transactions. Take care when using 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 pending work to be committed.

6.2 UTM Operations

This section describes the Oracle Database-specific points that you should consider when using UTM operations. The points in this section refer to PEND (Program Unit End) and RSET (Reset) operations. These operations represent the common synchronization point between openUTM and the Oracle Database.

When you issue a PEND call, UTM calls the Oracle Database internally 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

6.2.1 RSET and PEND RS

Resetting a UTM transaction implies rolling back the Oracle Database transaction.

6.2.2 PEND ER and PEND FR

When using these calls to terminate a UTM transaction, the Oracle Database transaction is also rolled back.

6.2.3 PEND KP, PEND PR, and PEND PA

These operations only end a UTM dialog step without affecting the corresponding Oracle Database transaction.

6.2.4 PEND RE, PEND FI, PEND SP, and PEND FC

These PEND calls cause an implicit COMMIT to be run. All cursors that have not been explicitly closed, are closed.

7

Globalization Support

This chapter describes the globalization support available with Oracle Database 11*g* Release 2 (11.2) for Fujitsu BS2000/OSD, with information about the following:

- Specifying a 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 (relating to 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:

```
011NLS.LXnnnnn.NLB
```

These files are for internal use only. You should not make changes to them. If you need a character set, language, or territory code that is not present, then contact your Oracle Support Services representative, who will be able to check whether any updates are available.

User-defined character sets as documented in the *Oracle Database Globalization Support Guide* are not supported for this release.

7.1 Specifying a Language, Territory, and Character Set

To choose the language, territory, and character set that you want to work with, you must carry out separate procedures for Oracle Database and the supported Oracle Database utilities.

7.1.1 Oracle Database

For the Oracle Database, your database administrator sets the `NLS_LANGUAGE` and `NLS_TERRITORY` parameters in the initialization files as described in the *Oracle Database Globalization Support Guide*.

7.1.2 Other Oracle Database Products

For the supported Oracle Database products, you can choose a language, territory, and character set from within BS2000/OSD by setting the value of the ORAENV variable NLS_LANG. Set this environment variable 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
```

7.2 Supported Language Conventions

Oracle Database 11*g* Release 2 (11.2) for Fujitsu BS2000/OSD provides support for language conventions, such as day and month names, 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`

7.3 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`

7.4 Supported Character Sets

Oracle Database 11*g* Release 2 (11.2) supports the following character sets for servers and clients under BS2000/OSD:

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
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)

Name	Description	Usage
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 should only be used as client character sets.

The character set WE8BS2000E must be used as database character set if you want to store the euro symbol in the database or if you want 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 listed in *Oracle Database Globalization Support Guide*.

Note: A Unicode database character set is not supported on BS2000/OSD. If you want to store Unicode characters in the database, then you must make use of Unicode datatypes `NCHAR`, `NVARCHAR2`, and `NCLOB`. During database creation you can specify either AL16UTF16 or UTF8 as the national character set for these datatypes. For more information about Unicode support, refer to Chapter 6 of the *Oracle Database Globalization Support Guide*.

7.5 Location of Message Files

All message files are located in `ORAMESG.LIB` under the installation user ID.

7.6 Linguistic Definitions

All the linguistic definitions listed in the "Globalization Support" chapter of *Oracle Database Globalization Support Guide* are available.

A

Oracle Error Messages for BS2000/OSD

This appendix lists the messages of Oracle Database 11*g* Release 2 (11.2) for Fujitsu BS2000/OSD together with possible causes and suggested actions. The messages shown in this chapter may be accompanied by additional text when displayed on screen. This text identifies the function that detects the problem, and can include internal status codes, or BS2000 system macro return codes, or both. These codes will help your Oracle Support Services Representative in determining the cause of a problem.

Sometimes, for example in the early stages of initialization when the message components are not available, the Oracle Database cannot issue a regular Oracle message. If this occurs, then the 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 your Oracle Support Services Representative.

ORA-05001: Unsupported BS2000 Version

Cause: The active version of the BS2000/OSD operating system is not supported by Oracle Database 11*g* Release 2 (11.2).

Action: Upgrade to a more recent BS2000/OSD 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: Make sure that your application program runs in 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, the 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. 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: Make sure that your application program runs in 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 (the corresponding ILCS routine returned the error).

Action: Make sure that the user address space is large enough (in the JOIN entry) and that there is no temporary memory saturation. Then re-run the program. If you need further help, then contact your Oracle Support Services Representative.

ORA-05005: Error: IT0INITS called in PROLOD

Cause: This is an internal error and should not occur.

Action: Contact your Oracle Support Services Representative.

ORA-05006: Error: sltga already initialized

Cause: The initialization routine for the `sltga` is called more than one time.

Action: Check if more than one stub module (`PROSTUB`, `XAOSTUB`) is linked to the application.

ORA-05007: failed to load OSNTAB

Cause: This message will normally be preceded by a BS2000 `BLS-`*`nnnn`* message. The most likely reason is that the `ORALOAD` library cannot be found.

Action: Contact your database administrator about the `ORALOAD` library. If you cannot identify the cause of the problem, then contact your Oracle Support Services Representative.

ORA-05008: failed to load requested network driver

Cause: This message will be normally preceded by a BS2000 `BLS-`*`nnnn`* message. The most likely reason is that the `ORALOAD` library cannot be found.

Action: Contact your database administrator about the `ORALOAD` library. If you cannot identify the cause of the problem, then contact your Oracle Support Services Representative.

ORA-05009: osnsgl: user connects invalid in kernel

Cause: A database link was set up using the single-task driver (S:). This is invalid, as the single-task driver can only be used for call connections on the user side.

Action: Select a different network driver for the database link.

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 your Oracle Support Services Representative.

ORA-05011: bad file size

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05012: bad block size

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05013: bad filename parse

Cause: A file name being analysed 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 blocksizes. This may indicate an internal error and should not normally occur.

Action: If you cannot identify the cause of the problem, then contact your Oracle Support Services Representative.

ORA-05015: text file open failed

Cause: An Oracle Database text or command 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-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 (for example a file may not be accessible for a cross-user ID single-task client).

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 your 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 your 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 your Oracle Support Services Representative.

ORA-05021: file read failed

Cause: An I/O 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 your 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 (block number < 1 or > file size). This is an internal error and should not normally occur.

Action: Contact your 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 your 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 your database administrator who may know why this error has occurred. If your database administrator cannot find the cause of the problem, then contact your Oracle Support Services Representative.

ORA-05027: file does exist

Cause: When attempting to create a new file, the file is found to exist and not be 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 your Oracle Support Services Representative.

ORA-05030: SID not defined

Cause: When the system identification 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 identification 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 your 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 Appendix B, "Oracle Environment Variables" for further information).

ORA-05034: bad seal

Cause: Internal error. An internal file control structure is found to be corrupt.

Action: Contact your 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 your 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 your 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: The Oracle Database kernel has been re-entered at the top. This should not happen.

Action: Make sure that your 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 your 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: Make sure that the user address space is large enough (in the `JOIN` entry) and that there is no temporary memory saturation. Then re-run the program. If you need further help, then contact your Oracle Support Services Representative.

ORA-05041: Interrupt in soarch

Cause: The archiver process was unexpectedly interrupted.

Action: If you cannot identify the cause of the problem, then contact your Oracle Support Services Representative.

ORA-05042: soarch: Buffer overflow

Cause: The archiver process detected an internal buffer overflow.

Action: If you cannot identify the cause of the problem, then contact your Oracle Support Services Representative.

ORA-05043: Archive control string too long

Cause: The archive control string is too long.

Action: Shorten this parameter and restart the database.

ORA-05044: Archive generated filename too long

Cause: The file name is generated from the values of the initialization parameters `log_archive_format` and `log_archive_dest`. This has resulted in a file name that is too long.

Action: Issue the command `ALTER SYSTEM ARCHIVE LOG START TO VALID_DEST` where `VALID_DEST` is a valid BS2000 file name.

ORA-05045: Archive file creation/open error

Cause: The archive file is normally allocated dynamically. Either this or the subsequent open failed. Possible causes are either insufficient space left on disk, or a bad archive file allocation parameter in `ORAENV` .

Action: Make sure that the optional `ORAENV` parameter is correct and that sufficient disk space is available.

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 your 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 your 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 your 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 DBA ORAENV file.

Action: Specify a correct value.

ORA-05054: invalid or missing PGA_SIZE

Cause: An invalid value for the PGA_SIZE environment variable has been specified in the DBA ORAENV file. You should never 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 your 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, then the default may be inappropriate for your case.

Action: Refer to the section "Address Space Planning" in the "Oracle Database System Architecture and Implementation" chapter of the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD* for further information.

ORA-05056: no more context space

Cause: During processing of a SQL request, 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 your application program (if the error occurred when you were using an application program) is not using excessive memory. Otherwise, contact your Oracle Support Services Representative.

ORA-05058: assert failed: SGA not mapped

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05059: assert failed: not in kernel

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05060: SGA not created

Cause: After you issued 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 identification is not being used for two different databases. Otherwise, contact your 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 identification, 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 your database administrator). Otherwise, contact your 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 hexadecimal value giving the full virtual address for the `SGA` memory pool. Correct the value and re-issue the `STARTUP` command.

ORA-05064: cannot allocate SGA

Cause: After creating the memory pool, the `REQMP` to allocate the space failed. Probable operating system error.

Action: If you cannot identify the cause of the problem, then contact your 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 your 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 the section on "Address Space Planning" in the chapter "Oracle Database System Architecture and Implementation" of the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD* , and adjust the relevant initialization file and `ORAENV` variables. Inspect the JOIN entry for your address space limit. Contact your System Administrator to find out about shared subsystems and their placement in the address space. Make sure 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 your 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 single-task, user task or a network server task.

Action: Check for such hanging tasks. Cancel these tasks, then restart the database.

ORA-05069: Unexpected SGA memory pool problem

Cause: The `ENAMP` macro returned an unexpected error code.

Action: Contact your Oracle Support Services Representative.

ORA-05070: cannot enable TPA ser.item

Cause: Probable operating system error.

Action: Contact your Oracle Support Services Representative.

ORA-05071: cannot ENQ on TPA ser.item

Cause: Probable operating system error.

Action: Contact your Oracle Support Services Representative.

ORA-05072: cannot enable post/wait item

Cause: Probable operating system error.

Action: Contact your Oracle Support Services Representative.

ORA-05073: error in post

Cause: An interprocess 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 your Oracle Support Services Representative.

ORA-05074: error in wait

Cause: An interprocess 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 your Oracle Support Services Representative.

ORA-05075: error in task table manager

Cause: Internal error.

Action: Contact your Oracle Support Services Representative.

ORA-05076: error setting spid

Cause: Probable operating system error.

Action: Contact your Oracle Support Services Representative.

ORA-05077: cannot enable HIA event

Cause: Probable operating system error. The HIA (Here I Am) event item is used during startup to communicate between a started background task and the starting SQL*DBA program.

Action: Contact your 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 any 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 your Oracle Support Services Representative.

ORA-05079: internal asynchronous IO error

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05101: bind-error xxxxxxxx for module/library

Cause: The Oracle Database/UTM attach module could not be loaded. One possible reason is that the Oracle Database has been installed under a user ID

different from $ORAC1120 and that the installation procedure has not executed correctly.

Action: Ensure that the ORAUID definition in the ORAENV file is correct. Otherwise, contact your Oracle Support Services Representative.

ORA-05102: module verification failure: ORADBCN@

Cause: The UTM application has probably been link-edited with an Oracle Database version different from the Oracle Database version used at execution.

Action: Re-link the UTM application. If the error persists, then contact your Oracle Support Services Representative.

ORA-05103: generated TSKM too short

Cause: The TSKM area was overwritten by the Oracle Database.

Action: Change the parameter LTHTSKM in the KDCDB/KDCDBO macro.

ORA-05104: generated TAM too short

Cause: The TAM area was overwritten by the Oracle Database.

Action: Change the parameter LTHTAM in the KDCDB/KDCDBO macro.

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 run 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 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 your Oracle Support Services Representative.

ORA-05111: error attaching to memory pool

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05112: error creating memory pool

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05114: bad pool base

Cause: An invalid value for the base address parameter of the shared pool, such as, COM_BASE has been specified in the ORAENV file.

Action: If this value is specified, then it must be a hexadecimal 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. Generally, this message will be preceded by a `BLS-`*`nnnn`* message from the operating system.

Action: Make sure that the `ORALOAD` link name identifies the correct `ORALOAD` library. Then restart the program. If you cannot identify the cause of the problem, then contact your Oracle Support Services Representative.

ORA-05117: cannot attach to socket subsystem

Cause: An application could not be bound to the sockets subsystem. Generally this message will be preceded by a `BLS-`*`nnnn`* message from the operating system.

Action: Contact your Oracle Support Services Representative.

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 your 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: Contact your Oracle Support Services Representative.

ORA-05120: waiting for shared module to be loaded timed out

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05121: waiting for initialization of shared module timed out

Cause: This is an internal error and should not normally occur.

Action: Contact your Oracle Support Services Representative.

ORA-05126: Missing IT0PCD address

Cause: The `ILCS` run-time link-library is probably missing.

Action: Contact your System Administrator.

ORA-05127: PARAM-LIST AT CALL ORACLE NOT OK

Cause: System error.

Action: Contact your Oracle Support Services Representative.

ORA-05128: COMMIT/ROLLBACK/CONNECT NOT ALLOWED IN UTM-PROGRAM

Cause: Illegal `SQL COMMIT/ROLLBACK/CONNECT` found in UTM program.

Action: Correct the UTM program accordingly.

ORA-05131: ORADBCO-Call not allowed

Cause: System error.

Action: Contact your Oracle Support Services Representative.

ORA-05132: TA for User x is committed by the Resource-Manager

Cause: Transaction has been committed before failing.

Action: None.

ORA-05133: No Connect-String in Startparams found

Cause: In the start parameters there must be at least one open string for the Oracle Database.

Action: Refer to the chapter "UTM Product Set" in the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD* and correct the start parameters.

ORA-05134: DBSTAT secondary opcode inconsistent

Cause: System error.

Action: Contact your Oracle Support Services Representative.

ORA-05135: Error x Recover PTC-list, Instance y

Cause: System error.

Action: Check if UTM is correctly installed and select privileges are granted to XA-tables. Refer to the chapter on openUTM, or contact the Oracle Support Services Representative.

ORA-05136: Maximum number of instances exceeded

Cause: The maximum number of open strings in the start parameters has been exceeded.

Action: Refer to the chapter "UTM Product Set" in the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD* and correct the start parameters.

ORA-05137: Error x at Open Instance y

Cause: The connection to instance y is not possible.

Action: Start the Instance with all the required servers. If the error persists, then contact your Oracle Support Services Representative.

ORA-05138: Error *x at Close Instancey*

Cause: Error occurred when disconnecting from instance y.

Action: Because the disconnection has just been done, there is no action.

ORA-05139: Error x at Start Transaction for User z, Instance y

Cause: The start of transaction in instance y is invalid.

Action: Restart the UTM transaction. If the error persists, then close the UTM application and do a restart.

ORA-05140: Error x at Continue Transaction for User z, Instance y

Cause: The continuation of transaction in instance y is invalid.

Action: Restart the UTM transaction. If the error persists, and the error is not 'XAER_PROTO' (this means, that the transaction is rolled back because of longlock), then shut down the UTM application and restart.

ORA-05141: Error x at Break Transaction for User z, Instance y

Cause: The break of transaction in instance y is invalid.

Action: Restart the UTM transaction. If the error persists, and there are no cursor operations in the preceded dialog step, then shut down the UTM application and do a restart.

ORA-05142: Error x at End Transaction for User z, Instance y

Cause: The end of a transaction in instance y is invalid.

Action: Restart the UTM transaction. If the error persists, then shut down the UTM application and do a restart.

ORA-05143: Mismatch in TA for User(s) z

Cause: In the UTM warmstart there is a mismatch between openUTM, or between the Oracle Database instance(s), or both.

Action: Clear the Oracle Database instance(s), create new KDCDEF for UTM and restart the UTM application.

ORA-05144: Error x at Prepare Transaction for User z, Instance y

Cause: The preparation for committing a transaction in instance y is invalid.

Action: Restart the UTM transaction. If the error persists, then shutdown the UTM application and do a restart.

ORA-05145: Error x at Commit Transaction for User z, Instance y

Cause: Attempt to commit transaction in instance y unsuccessful.

Action: Restart the UTM transaction. If the error persists, then shutdown the UTM application and restart.

ORA-05146: Error x at Rollback Transaction for User z, Instance y

Cause: The rollback of transaction in instance y is invalid.

Action: No action, but if the error persists, then shutdown the UTM application and restart.

ORA-05147: TA for User z committed; Reason: Recovery

Cause: In a openUTM warmstart an interrupted transaction has been committed.

Action: None.

ORA-05148: TA for User z heuristic rolled back in Instance y

Cause: In a UTM warmstart an interrupted transaction has just been rolled back from the Oracle Database.

Action: Restart the UTM transaction.

ORA-05149: TA for User z rolled back; Reason: Internal Event

Cause: The end- or prepare-call was invalid. Therefore the transaction must be rolled back.

Action: Restart the UTM transaction.

ORA-05150: KDCS-PEND before DBFITA missing

Cause: System error.

Action: Contact your Oracle Support Services Representative.

ORA-05151: KDCS-PEND before DBPETA missing

Cause: System error.

Action: Contact your Oracle Support Services Representative.

ORA-05152: Linked Resource-Manager is not CAE-compatible

Cause: System error.

Action: Contact your Oracle Support Services Representative.

ORA-05153: xa_switch definition not found for specified Resource-Manager: s

Cause: System error.

Action: Contact your Oracle Support Services Representative.

ORA-05154: Syntax error in start parameters for Resource-Manager: s

Cause: In the start parameters for the Oracle Database there is a syntax error.

Action: Refer to the chapter "UTM Product Set" in your *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD* and correct the start parameters.

ORA-05155: Internal Error: malloc in dbstpa

Cause: System error (one possible reason: out of memory)

Action: Contact your Oracle Support Services Representative.

ORA-05156: Internal Error: realloc in dbstpa

Cause: System error (one possible reason: out of memory)

Action: Contact your Oracle Support Services Representative.

ORA-05157: Internal Error: malloc in up_recovery

Cause: This error message indicates a system error. One of the possible reasons for this issue is that the system does not have sufficient memory.

Action: Contact your Oracle Support Services Representative.

ORA-05158: IUTMDB-Function not supported

Cause: openUTM system error.

Action: Contact your Oracle Support Services Representative.

ORA-05159: TA for User(s) z rolled back; Reason: Recovery

Cause: In a openUTM warmstart one or more interrupted transactions have been rolled back.

Action: None.

ORA-05161: TCP/IP can't perform asynchronous test on break socket.

Cause: Select on break socket failed.

Action: Contact your System Administrator about TCP/IP networking problems. If the error persists, then contact your Oracle Support Services Representative.

ORA-05165: function not supported

Cause: Either Oracle Database 11*g* Release 2 (11.2) or BS2000/OSD 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 your 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. This could be caused by a missing `ORAENV` file or a missing `ORASID` entry in that file.

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 specify this variable, as the default value will be correct. Contact your 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 hexadecimal 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 the section on "Address Space Planning" in the chapter "Oracle Database System Architecture and Implementation" of the *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*, and adjust the relevant initialization file and ORAENV parameters. Inspect the JOIN entry for your address space limit. Contact your 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 your 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: This problem can be caused when you run a program in 24-bit mode and try to connect a single-task to a kernel residing above the 16MB line (because the database itself is running in 31-bit mode). If this is the cause of the error, then you must access the database in two-task mode (through SQL*Net). Refer to the *ENAMP macro description* in the BS2000 documentation for other possible reasons. If you cannot identify the cause of the problem, then contact your 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 contact your database administrator.

ORA-05180: Cannot load character set table

Cause: One of the modules containing character set tables cannot be loaded.

Action: Verify that the ORALOAD library is accessible through LINK=ORALOAD. For further information about link names and the ORALOAD library, Refer to Chapter 1, "Getting Started" in this manual. If you cannot identify the cause of the problem, then contact your Oracle Support Services Representative.

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 will normally be preceded by a BS2000 BLS-*nnnn* message), or the loaded module is incompatible with the version of the stub module.

Action: Make sure the ORALOAD link name exists and points to the current ORALOAD library. Re-link your 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 identification, ORASID in the ORAENV file.

Action: Ensure that the ORAENV file definition is correct. Otherwise contact your 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 part of the address range needed for the memory pool is already allocated by the user program.

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 your 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 your 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: Make sure 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 your 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 your 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; 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 your 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 will explain the error. The system will cause the program execution to stop (TERM ABNORMAL with DUMP will be shown).

Action: If you cannot identify the cause of the problem, then contact your Oracle Support Services Representative.

B

Oracle Environment Variables

This appendix describes variables that can be specified in the ORAENV file or the POSIX shell. Oracle parameters, such as ORACLE_SID and NLS_LANG, may be specified in the ORAENV file 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. In the POSIX shell, you must follow the UNIX rules to set and export the environment variables.

The following table describes the variables that are categorized into three classes.

Class	Description
DBA	These variables are for database administration purposes. Most DBA variables are evaluated only during database startup.
USER	These variables can be specified by ordinary users as well as by the DBA. When these variables are specified in a particular user's ORAENV file, they modify that user's environment only.
NET	These variables apply to Oracle Net Services components. These variables should be included in the ORAENV file of the DBA.

The class (or classes) to which a variable belongs is noted in the variable descriptions in this appendix.

Any DBA or NET variables specified in an ordinary user's ORAENV file are ignored.

B.1 ORAENV Rules

You should consider the following general rules when writing 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.
- Since the variable list is conceptually open ended, errors in variable names are not recognized. This means that the value of any variable whose name is mis-typed 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 a variable is actually used, not when it is read from the ORAENV file.

  For example:

  ```
  ORAUID=$ORAC1120
  SQLPATH=&ORAUID..RDBMS.ADMIN
  ```

  assigns the value $ORAC1120.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 last occurrence is used.
- If no value is given 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:

B.2.1 LOGNAME

The LOGNAME variable always contains the current BS2000 user ID. You cannot 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 always contains the last part of the current START_PROGRAM program name. 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, and defaults to '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, day-to-day use by Oracle DBAs and users.

B.3.1 CLN_BASE

This variable specifies the address of the shared code pool of customer written database applications for CORE, NLS, and NET.

Format:

CLN_BASE=*address*

Classification:

USER

Default:

CLN_BASE=37M

B.3.2 CLN_MPID

This variable specifies the identification of the shared code pool of customer written database applications for CORE, NLS, and NET.

Format:

CLN_MPID=*sid*

Classification:

USER

Default:

CLN_MPID=*&ORASID*

B.3.3 DEFAULT_CONNECTION

This variable provides a default host string for connect requests where no host string is specified. If you always connect to the same database, then it may be convenient to specify this. This value should contain everything you would otherwise specify after the "@" character.

Format: DEFAULT_CONNECTION=*host-string*

Classification: USER

Example:

```
DEFAULT_CONNECTION=TNS:
(DESCRIPTION=
(ADDRESS=
(PROTOCOL=TCP)
(HOST=MADRID)
(PORT=1521))
(CONNECT_DATA=
(SERVICE_NAME=PROD)))
```

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

Classification: USER

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

Classification: USER

Default: IMP_CLIB_FILE_IO=TRUE

B.3.6 NLS_LANG

This variable specifies the default language and character set. For example:

```
NLS_LANG=GERMAN_GERMANY.D8BS2000
```

Format: NLS_LANG=*language_territory.character-set*

Classification: USER, DBA

Default: NLS_LANG=AMERICAN_AMERICA.WE8BS2000

B.3.7 OPS_JID

This variable is used for concatenation with the OS_AUTHENT_PREFIX, refer to 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, /.jobname LOGON..., is used instead. This is useful when many users are sharing one BS2000 user ID.

Format: OPS_JID=*userid/jobname*

Classification: DBA

Default: *userid*

B.3.8 ORADUMP

This variable specifies the dump file for Oracle Database and user trace output.

Format: ORADUMP=*dump-file*

Classification: USER, DBA

Default: ORADUMP=OTRC.?.&TSN..&PGM..TRC

Example:

```
ORADUMP=(SYSOUT)
```

This assignment redirects the trace output to SYSOUT.

B.3.9 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)

Classification: USER, DBA

> **Note:** Oracle recommends that you use the ORACLE_SID variable.

B.3.10 PRINTPAR

This variable specifies optional variables for the /PRINT command issued for SPOOL OUT spool files. 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 temp.spoolfile,&PRINTPAR
```

Format: PRINTPAR=*print-options*

Classification: USER

B.3.11 SQLPATH

This variable specifies a path where SQL*Plus looks for command files. Elements of the path are separated by semicolons (;). For example:

```
SQLPATH=PRIVATE;$ORAC1120
```

This assignment will cause SQL*Plus to look for filename.SQL, then for PRIVATE.*filename*.SQL, and finally for $ORAC1120.filename.SQL.

Format: SQLPATH=*search-path*

Classification: USER, DBA

B.3.12 SSSIDPWF

This variable specifies the password file for remote instance start. For further information, refer to "Administering Oracle Database" in *Oracle Database Installation and Administration Guide for Fujitsu BS2000/OSD*.

Format: SSSIDPWF=*password-file*

Classification: DBA

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 contained in the initialization file.

Oracle recommends that database startup and shutdown, background jobs, and network jobs should all refer to the same ORAENV file to ensure that the variables are consistent.

Note that the default values listed in the following section are built-in defaults, most of them are over-ridden by settings in the shipped DEMO.P.ORAENV.

B.4.1 Address and Size Specification

Several of the variables described in this section define memory addresses and sizes. The notation used to specify these items is 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 all set the KNL_BASE variable to 8M:

```
KNL_BASE=8M
KNL_BASE=8388608
KNL_BASE=X'800000'
```

B.4.2 BGJPAR

This variable specifies the parameters for the ENTER-PROCEDURE command used when starting background jobs. The ENTER-PROCEDURE command is used to submit jobs as follows:

```
.jobname ENTER-PROCEDURE jobfile, &BGJPAR
```

Format: BGJPAR=*parameters*

Classification: DBA

> **Note:** The BGJPAR variable is set up by the installation procedure.

B.4.3 BGJPRC_UID / BGJPRC_SID

These variables specify the user ID and orasid of the file for the background enter jobs. If the use of a special enter job file is desired, then the parameters must be set to the desired userid and orasid.

Format:

```
BGJPRC_UID=$userid
BGJPRC_SID=sid
```

Classification: DBA, NET

Default:

```
BGJPRC_UID=&ORAUID
BGJPRC_SID=DEMO
```

B.4.4 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

Classification: DBA,USER, NET

Default: BGJ_LOG_JOBSTART=N

B.4.5 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 at the most 4 alphanumeric characters

parameters is the parameters for the ENTER-PROCEDURE command as described in the BS2000/OSD commands

Classification: DBA,USER, NET

B.4.6 sid_USER

This variable specifies the USER-ID where the instance assigned by *SID* resides.

Format: *sid*_USER=*userid*

Syntax: *sid* is a string of at the most 4 alphanumeric characters

userid is a string of at most 8 alphanumeric characters which follows the rules of a BS2000/OSD USER-ID

Classification: DBA,USER, NET

B.4.7 user_ACCOUNT/ user_PASSWORD

*user*_ACCOUNT or *user*_PASSWORD define credentials of a USER-ID, which are used by the ENTER-PROCEDURE command to start a process.

Format: *user*_ACCOUNT=*account*

*user*_PASSWORD=*password*

Syntax: *user* is a string of at the most 8 alphanumeric characters, which follows the rules of a BS2000/OSD USER-ID and must match a USER-ID defined by the parameter *sid*_USER.

account is a string of at the most 8 alphanumeric characters, which follows the rules for a BS2000/OSD account number.

password is a string of at the most 8 alphanumeric characters, which follows the rules for a BS2000/OSD password.

Classification: DBA, NET

B.4.8 COM_MPID

This parameter specifies the identification of the shared code pool of the Oracle instance for CORE, NLS, and NET.

Format: COM_MPID=*sid*

Classification: DBA

Default: COM_MPID=*&ORASID*

B.4.9 COM_BASE

This parameter specifies the address of the shared code pool of the Oracle instance for CORE and NLS.

Format: COM_BASE=*address*

Classification: DBA

Default: COM_BASE=*37M*

B.4.10 JOBID

This variable is used internally in identifying the background tasks and generating task-specific names. You see it in some places, but you should never specify it yourself.

Classification: DBA

B.4.11 KNL_BASE

This variable gives the base address where the shared memory pool is mapped in memory. This must be an integral number of megabytes.

Format: KNL_BASE=*address*

Classification: DBA

Default: KNL_BASE=72M

B.4.12 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*

Classification: DBA, USER

B.4.13 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 64KB boundary.

Format: PGA_BASE=*address*

Classification: DBA

Default: PGA_BASE=189M

Note: The value of PGA_BASE is taken from the kernel if the shared kernel is already loaded.

B.4.14 PGA_SIZE

This variable specifies the size of the fixed part of the PGA. This variable should not be changed from its default value.

Format: PGA_SIZE=*size*

Classification: DBA

Default: PGA_SIZE=64K

Note: The value of PGA_SIZE is taken from the kernel if the shared kernel is already loaded.

B.4.15 SF_PBLKSIZE

This variable specifies the physical blocksize of redo log files.

Format: SF_PBLKSIZE=2K|4K

Classification: DBA

Default: 2K

> **Note:** This variable cannot be changed after database creation. Once you specify a value different from the default, you must specify it in all future calls.

B.4.16 SGA_BASE

This variable gives the address where the SGA is mapped into memory, and must represent a megabyte-boundary.

Format: SGA_BASE=*address*

Classification: DBA

Default: SGA_BASE=190M

> **Note:** The value of SGA_BASE is read from the kernel if the shared kernel is already loaded. 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:

B.5.1 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

Classification:

DBA, USER, NET

Default:

BREAK_HANDLING=ON

B.5.2 TNS_ADMIN

This variable specifies the user ID of the Oracle Net Services configuration files, for example, LISTENER.ORA, TNSNAMES.ORA and SQLNET.ORA. If TNS_ADMIN is not defined, then the configuration files are searched under the local user ID with the prefix NETWORK.ADMIN.

Format: TNS_ADMIN=$userid

Classification: DBA, USER, NET

B.5.3 TNS_BEQ_TIMEOUT

This variable specifies the time after which a connection between a parent and a child process is closed if there is no communication between them.

Format: `TNS_BEQ_TIMEOUT=`*`lifetime`* (in seconds)

Classification: `NET`

Default: `TNS_BEQ_TIMEOUT=180`

B.5.4 TNS_UPDATE_IPNODE

This variable forces the Oracle Net software to change always the server's IP-Node name to an IP-Node address.

Format: `TNS_UPDATE_IPNODE=TRUE/FALSE`

Classification: `NET`

Default: `TNS_UPDATE_IPNODE=FALSE`

Index

Symbols

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

R

S

T

U