Oracle® Transparent Gateway for DB2

Installation and User's Guide

10*g* Release 2 (10.2) for IBM z/OS (OS/390)

B16220-02

September 2006

ORACLE®

Oracle Transparent Gateway for DB2 Installation and User's Guide, 10*g* Release 2 (10.2) for IBM z/OS (OS/390)

B16220-02

Primary Author: Anne Bers and Oanh Duong

Contributing Author: Sutapa M. Kuthiala and Maitreyee Chaliha

Contents

6 Oracle Net

7 Administering the Gateway

8 Using the Gateway

9 Developing Applications

10 Error Messages, Diagnosis, and Reporting

11 Migration and Coexistence with Existing Gateways

Preface

This guide provides instructions for administering and configuring Oracle Transparent Gateway for DB2 for 10*g* Release 2 (10.2) for z/OS.

Intended Audience

Read this guide if you are responsible for performing such tasks as:

- Installing and configuring the Oracle Transparent Gateway for DB2
- Administering the gateway
- Using the gateway

Understand the fundamentals of z/OS operating systems before using this guide. This guide provides only information on Oracle products and their interactions with z/OS.

Product Name

The complete name for this product is Oracle Transparent Gateway for DB2 for z/OS. To maintain readability and conciseness in this document, the Oracle Transparent Gateway for DB2 for z/OS may be referred to as the Gateway, the Oracle Transparent Gateway, the DB2 Gateway, or TG4DB2.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of

assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

The Oracle Documentation Set

The documentation set has two parts: the documentation specific to the Oracle Transparent Gateway for DB2, and general gateway documentation. Your site automatically receives both documentation parts for the Oracle products that you have purchased. Use the general gateway documentation to learn about gateway concepts, and use the Oracle Transparent Gateway for DB2 documentation to learn how to install, administer, and use this gateway.

The Oracle Transparent Gateway for DB2 documentation consists of the following documents:

- *Oracle Transparent Gateway for DB2 Installation and User's Guide for IBM z/OS (OS/390)*
    - *Oracle Database Heterogeneous Connectivity Administrator's Guide*

- *Oracle Database Installation Guide for IBM z/OS (OS/390)*
- *Oracle Database Messages Guide for IBM z/OS (OS/390)*

Related Documents

There are two parts to the documentation set: platform-specific documentation and product-specific documentation. Your site automatically receives both for the Oracle products you have purchased. Use the product-specific documentation to learn how to use a product, and use the platform-specific documentation to learn about special requirements or restrictions for using that product under z/OS.

Platform-Specific Documentation

The z/OS-specific documentation set is used to install, maintain, and use Oracle Database 10*g* for z/OS products, and consists of the following documents:

- *Oracle Database Installation Guide for IBM z/OS (OS/390)*
- *Oracle Database Messages Guide for IBM z/OS (OS/390)*
- *Oracle Database Release Notes for IBM z/OS (OS/390)*
- *Oracle Database System Administration Guide for IBM z/OS (OS/390)*
- *Oracle Database User's Guide for IBM z/OS (OS/390)*

Product-Specific Documentation

Product-specific documentation describes how to use the Oracle Database 10*g* products. The information in these books is constant for all operating systems under which the products run. The following product-specific documents are referenced in this guide:

- *Oracle Database User's Guide for IBM z/OS (OS/390)*
- *Oracle Database Advanced Security Administrator's Guide*
- *Oracle Database Heterogeneous Connectivity Administrator's Guide*
- *Oracle Database SQL Reference*
- *Oracle Database Net Services Reference*
- *Oracle Streams Concepts and Administration*
- *Oracle Call Interface Programmer's Guide*
- *Programmer's Guide to the Oracle Precompilers*

- *SQL*Plus User's Guide and Reference*

IBM Documentation

The following IBM documents are referenced in this guide:

- *MVS Initialization and Tuning Reference*
- *MVS Systems Commands*
- *RACF Command Language Reference*
- *RACF System Administrator's Guide*

SQL*Plus Prompts

The SQL*Plus prompt, SQL>, appears in SQL statement and SQL*Plus command examples. Enter your response at the prompt. Do not enter the text of the prompt (SQL>) in your response.

Storage Measurements

Storage measurements use the following abbreviations:

- K, for kilobyte, which equals 1,024 bytes
- M, for megabyte, which equals 1,048,576 bytes
- G, for gigabyte, which equals 1,073,741,824 bytes

Conventions Used in this Guide

The following typographic conventions are used in this guide:

Convention	Meaning
italics	Indicates a variable, including variable portions of file names. It is also used for emphasis and book titles
UPPERCASE	Indicates batch and TSO commands and file names, SQL commands, reserved words, keywords, initialization parameters, and environment variables
`monospace`	Indicates z/OS UNIX System Services shell commands and file names, code examples, directory names, and user names

Convention	Meaning
oracle_hlq	Is the standard example for a high-level data set name qualifier. Substitute your system's actual high-level qualifier, for example, ORACLE.V10G. These qualifiers may appear in lowercase or UPPERCASE typeface
<> Angle brackets	Indicate that the enclosed arguments are required and at least one of the arguments must be entered. (Do not enter the brackets, themselves.)
[] Square brackets	Indicate that the enclosed arguments are optional. (Do not enter the brackets, themselves.)
{} Braces	Indicate that one of the enclosed arguments is required. (Do not enter the braces, themselves.)
| Vertical lines	Separate the choices
... Ellipses	Indicate that the preceding item can be repeated. You can enter an arbitrary number of similar items
Other punctuation	Must be entered as shown unless otherwise specified. For example, commas and quotes

Switches

Switches are listed with a brief description and any z/OS-specific information. Refer to the product-specific documentation for syntax and for a thorough description of its purpose. Before using a command line option switch, such as -A or -C, refer to the appropriate product-specific documentation.

1 Introduction

In today's fast-paced global economy, information is a company's most valuable resource. Companies often have a variety of applications and data that are geographically scattered and use incompatible networks, platforms, and storage formats. In a single company, data is likely to be dispersed across several systems and hundreds of desktops. Oracle Open Gateways simplify complex systems and remove obstacles to information mobility.

This chapter includes the following sections:

Oracle Open Gateways use the facilities of the Oracle Database 10*g* heterogeneous services. For further information about Heterogeneous Services, refer to the *Oracle Database Heterogeneous Connectivity Administrator's Guide*.

Version 10 Gateways

The Oracle Database 10*g* database server continues to improve its distributed database facilities that make the integration of enterprise data practical. Data can be accessed remotely using both SQL and PL/SQL procedure calls in a transparent

manner, as if the data were local. Also, data can be stored in both Oracle Database 10*g* database servers and non-Oracle servers.

Version 10 gateways are tightly integrated with the Oracle Database 10*g* database server, enabling improved performance and enhanced functionality while still providing transparent integration of Oracle and non-Oracle data. For example, connection initialization information is available in the local Oracle database server, reducing the number of round trips and the amount of data sent over the network. Running SQL is also faster, because statements issued by an application are parsed and translated once and can then be reused by multiple applications.

Version 10 gateways leverage the enhancements in the Oracle Database 10*g* database server, and you can quickly extend those benefits to your non-Oracle data.

Advantages of the Gateway

Oracle Transparent Gateway for DB2 enables Oracle client applications to access DB2 through Structured Query Language (SQL). The gateway and Oracle database server together create the appearance that all data resides on a local Oracle database server, even though data might be widely distributed. If data is moved from a DB2 database to an Oracle database server, then no changes in client application design or function are needed. The gateway handles all differences in data types and SQL functions between the application and the database.

Oracle Transparent Gateway for DB2 gives you the power to integrate your heterogeneous system into a single, seamless environment, enabling you to make full use of existing hardware and applications throughout your corporate-wide environment. You can eliminate the need to rewrite applications for each configuration and avoid the tedious, error-prone process of manual data transfer. Together with the Oracle tools, networking, and data server technology, Oracle Transparent Gateway for DB2 provides seamless, enterprise-wide information access.

Transparency at All Levels

By using Oracle Transparent Gateway for DB2, you can achieve transparency at every level within your enterprise.

- Location transparency

  Users can access tables by name, without having to understand the physical location of the tables.

- Network transparency

The gateways exploit Oracle Net, enabling protocol independence. Regardless of the quantity or type of communication protocols used, data access is seamless. Protocols supported for Oracle Net include IBM TCP/IP High Performance Native Sockets (HPNS), and Oracle Net cross memory driver.

- Operating system transparency

  You can access data stored under multiple operating systems without being aware of the operating systems that hold the data.

- Data storage transparency

  Data can be accessed regardless of the database or file format.

- Access method transparency

  You can use a single dialect of SQL for any data store, eliminating the need to code for database-specific access methods or SQL implementations.

Extension of Database Services

Following are some of the more sophisticated Oracle Database 10*g* database server services available through the gateway.

- SQL functions

  Your application can access all your data using Oracle SQL that is rich in features. Advanced Oracle Database 10*g* database server functions, such as outer joins, are available even if the target data stores do not support them in a native environment. The manner in which the gateways are integrated with the Oracle Database 10*g* database server ensures that the latest features of each database release are always available immediately to the gateway.

- Distributed capabilities

  Heterogeneous data can be integrated seamlessly because Oracle distributed capabilities, such as JOIN and UNION, can be applied against non-Oracle data without any special programming or mapping.

- Distributed query optimization

  The Oracle Database 10*g* database server can use its advanced query optimization techniques to ensure that SQL statements are run efficiently against any of your data. The data distribution and storage characteristics of local and remote data are considered equally.

- Two-phase commit protection

The Oracle two-phase commit mechanism provides consistency across data stores by ensuring that a transaction that spans data stores is still treated as a single unit of work. Changes are not committed, or permanently stored, in any data store unless the changes can be committed in all data stores that are affected.

- Stored procedures and database triggers

  The same Oracle stored procedures and database triggers can be used to access all your data, ensuring uniform enforcement of business rules across the enterprise.

Extension of Advanced Networking, Internet, and Intranet Support

The gateway integration with the Oracle Database 10*g* database server extends to non-Oracle data, the benefits of the Oracle internet and Oracle Net, and the Oracle client/server and server/server connectivity software. These powerful features include:

- Application server support

  Any Internet or intranet application that can access data in the Oracle database server can also incorporate information from data stores accessible through the gateways. Web browsers can connect to the Oracle database server using any application server product that supports Oracle software.

- Advanced security

  Non-Oracle data can be protected from unauthorized access or tampering during transmission to the client. This is done by using the hardware-independent and protocol-independent encryption and CHECKSUM services of the Advanced Networking Option (ANO).

- Wireless communication

  Oracle Mobile Agents, an Oracle industry-leading mobile technology, enables wireless communication to Oracle Database 10*g* database servers or any databases accessible through the gateways. This gives field personnel direct access to enterprise data from mobile laptop computers.

Dynamic Dictionary Mapping

The simple setup of the gateway does not require any additional mapping. Before an application can access any information, the application must be told the structure of the data, such as the columns of a table and their lengths. Many products require administrators to manually define this information in a separate data dictionary

stored in a hub. Applications access information using the hub dictionary instead of the native dictionaries of each database. This approach requires a great deal of manual configuration and maintenance. Administrators must update the data dictionary in the hub whenever the structure of a remote table is changed.

Inefficient duplication is not necessary with Oracle Transparent Gateway for DB2. The gateway uses the existing native dictionaries of each database. The applications access data using the dictionaries designed specifically for each database, which means that no redundant dictionary ever needs to be created or maintained.

SQL

Oracle Transparent Gateways ease the application development and maintenance by allowing you to access any data using a uniform set of SQL. Changes to the location, storage characteristics, or table structure do not require any changes to your applications. ANSI and ISO standard SQL are supported, along with powerful Oracle extensions.

Data Definition Language

Oracle Applications can create tables in target data stores by using native data definition language (DDL) statements.

Data Control Language

You can use native data control language (DCL) statements from an Oracle database server environment, allowing central administration of user privileges and access levels for heterogeneous data stores.

Passthrough and Native DB2 SQL

Running of native DB2 SQL can be passed through the gateway for running directly against DB2. This enables applications to send statements, such as a DB2 CREATE TABLE, to the gateway to run on a target DB2 system.

Stored Procedures

The gateway enables you to exploit both Oracle and non-Oracle stored procedures, leveraging your investments in a distributed, multi-database environment. Oracle stored procedures can access and update multiple data stores easily, without any special coding for the heterogeneous data access.

Oracle Stored Procedures

Oracle stored procedures enable you to access and update DB2 data using centralized business rules stored in the Oracle Database 10*g* database server. Using Oracle stored procedures can increase the database performance by minimizing network traffic. Instead of sending individual SQL statements across the network, an application can send a single EXECUTE command to begin an entire PL/SQL routine.

Native DB2 Stored Procedures

The gateway can run DB2 stored procedures using standard Oracle PL/SQL. The Oracle application runs the DB2 stored procedure as if it were an Oracle remote procedure.

Native DB2 User Defined Functions

DB2 User Defined Functions (UDFs) can be coded in Oracle SQL statements or as function calls in PL/SQL. The Oracle database calls the DB2 UDF to perform the function passing constants, column values or bind variables. The results of the UDF are then returned into the SQL statement or PL/SQL statement for processing. This feature allows you to leverage DB2 UDF development work and make those DB2 functions available to Oracle SQL and PL/SQL programs as if they were Oracle functions.

Applications

Any application or tool that supports the Oracle Database 10*g* database server can access over 30 different data sources through the Oracle gateways. A wide variety of open system tools from Oracle and third-party vendors can be used, even if the data is stored in legacy, proprietary formats. Hundreds of tools are supported, including improvised query tools, web browsers, turnkey applications, and application development tools.

Oracle Developer

Use Oracle Developer to build applications that can manipulate data stored in your DB2 database and Oracle database server. This includes designing forms, producing graphic displays, and creating a wide range of objects.

Oracle Discoverer

Use Oracle Discoverer to analyze, manipulate, and copy data residing in your DB2 database. This product gives you access to corporate data using a powerful data analysis tool.

SQL*Plus

Use SQL*Plus for moving data between the databases. This product gives you the ability to copy data between your department databases and corporate Oracle database servers.

Oracle Database Server Technology and Tools

The gateway is integrated into the Oracle database server technology, which provides global query optimization, transaction coordination for multisite transactions, support for all Oracle Net configurations, and so on. Tools and applications that support the Oracle database server can be used to access heterogeneous data through the gateway.

Two-Phase Commit and Multisite Transactions

The gateway can participate as a partner in multisite transactions and two-phase commit. How this occurs depends on the capabilities of the underlying data source, meaning that the gateway can be implemented as any one of the following:

- A full two-phase commit partner
- A commit point site
- A single-site update partner
- A read-only partner

The deciding factors for implementing the gateway are the locking and transaction-handling capabilities of your target database.

Oracle Transparent Gateway for DB2, by default, is configured as a commit point site (that is, commit confirm protocol). Optionally, you can configure the gateway as read-only if you choose to enforce read-only capability through the gateway. Other protocols are not supported.

Site Autonomy

All Oracle database server products, including gateways, supply site autonomy. For example, administration of a data source remains the responsibility of the original system administrator. With site autonomy implemented, gateway products do not override the security methods of the data source or operating environment.

Migration and Coexistence

The integration of a data source through the gateway does not require any changes to be made to applications at the data source. As a result of this, the Oracle database server technology is non-intrusive, providing coexistence and an easy migration path.

Security

The gateway does not bypass existing security mechanisms. Gateway security coexists with the security mechanisms already used in the operating environment of the data source.

Protection of Current Investment

With the gateway, you can continue to develop your information systems without foregoing your investments in current data and applications. Access your Oracle and DB2 data with a single set of applications and continue to use existing IBM applications to access your IBM data. You can also use more productive database tools and move to a distributed database technology without giving up access to your current data.

If you want to migrate to Oracle database server technology and productivity, then the gateway allows you to control the pace of your migration. As you transfer applications from your previous technology to the Oracle database server, you can use the gateway to move the DB2 data into Oracle database servers.

Gateway Architecture

The gateway architecture consists of three main components.

- Oracle Database Server

  The Oracle database server is an Oracle instance functioning as a client to access the gateway.

- Gateway

  Oracle Transparent Gateway for DB2 must be installed on a z/OS system.

- DB2 server

  The DB2 database server is the one being accessed by the gateway. IBM terminology for a DB2 server is DB2 Server for z/OS or DB2 Universal Database (UDB) for z/OS.

Multiple Oracle database servers can access the same gateway. A single gateway installation can be configured to access one DB2 server. Any number of DB2 gateways can access the same DB2 server.

How the Gateway Works

The gateway has no database functions. The gateway provides an interface by which the Oracle database server can direct SQL operations to a DB2 database.

Using a database link, the gateway is identified to the Oracle database server. The database link is the same construct used to identify other Oracle database servers.

Tables on the DB2 server are referenced in SQL as:

```
table_name@dblink_name
```

or

```
owner.table_name@dblink_name
```

If you create synonyms or views in the Oracle database server, then you can refer to tables on the DB2 server using simple names as though the table were local to the Oracle database server.

When the Oracle database server encounters a reference to a table on the DB2 server, the applicable portion of the SQL statement is sent to the gateway for processing. Any host variables associated with the SQL statement are bound to the gateway and, therefore, to the DB2 server.

The gateway is responsible for sending these SQL statements to the DB2 server. The DB2 server is responsible for running the SQL statements and for fielding and returning responses.

2

Release Information

This chapter describes the changes and corrected problems in this release. It includes the following sections:

Product Set

The following table lists the versions of the components and utilities included with this product.

Product	Release Number
Oracle Transparent Gateway for DB2	10.2.0.2.0
Oracle Net	10.2.0.2.0

Changes and Enhancements

This section lists the changes and enhancements to TG4DB2 by the release in which they were introduced.

Changes and Enhancements in Release 8.1.7

The following changes and enhancements were incorporated in the 8.1.7 release:

Auto Registration

Oracle Transparent Gateway for DB2 supports Heterogeneous Services auto registration. This feature will significantly speed session initialization by storing class capabilities in the server data dictionary rather than uploading the capabilities and storing them in memory for each session.

DB2 Version 6.1 and 7.1 Stored Procedures

Oracle Transparent Gateway for DB2 now supports IBM's new procedure for running store procedures in DB2 version 6.1 and 7.1.

Changes and Enhancements in Release 9.2.0

The following changes and enhancements were incorporated in the 9.2.0 release:

Operating System Dependent Interface (OSDI)

Like Oracle Database 10*g* for z/OS, Version 9.2.0.1.1 of the Oracle Transparent Gateway for DB2 is based on Operating System Dependent Interface (OSDI), an execution environment for Oracle products on z/OS, which was introduced with Oracle8*i*, Release 3 (8.1.7).

OSDI represents a significant change from the old subsystems in terms of how Oracle products interact with the z/OS operating system. These changes do not generally affect Oracle product behavior and the interfaces used by customer applications, but they do affect the installation, configuration, and administration of these products on z/OS.

If you are already an Oracle for z/OS customer with existing MPM or TNS subsystems, refer to Chapter 11, "Migration and Coexistence with Existing Gateways" for more information about differences between MPM and OSDI, and for migration and upgrade considerations.

TNSNAMES SID PARAMETER

Unlike previous MPM releases of the Oracle Transparent Gateway for DB2, the port number in the TNSNAMES entry is no longer sufficient to determine the destination of a connection request. The NET listener can now listen on a single port for multiple z/OS destination address spaces. The TNSNAMES entry must now include a `SID=` parameter. This parameter value must match the SID(xxxx) value in the OSDI service definition.

SQL*Plus Describe Support

Using RDBMS 9.2 as integrating server, you can now use the SQL*Plus `Describe` command to describe DB2 objects.

DB2 (var)graphic to Oracle (var)char(2) Mapping Support

This release of Oracle Transparent Gateway for DB2 provides DB2 `(var)graphic` to Oracle `(var)char(2)` mapping support. Refer to Appendix 12, "Globalization Support" for details.

Streams Heterogeneous Replication

Heterogeneous Replication is a new Oracle Database feature using Oracle gateway technology. You need a minimum release of 9.2.0.2 database server release to take advantage of the feature. Refer to "STREAMS Replication" on page 8-10, for details.

Changes and Enhancements in Release 10.1.0.2.0

The following changes and enhancements were incorporated in the 10.1.0.2.0 release:

RRSAF (Recoverable Resource Manager Services Attachment Facility)

All releases of Oracle Transparent Gateway for DB2 prior to 10*g* used the DB2 Call Attachment Facility (CAF) to access DB2. With this release, the DB2 RRSAF is now used for all DB2 access.

This is an improved interface and simplifies installation and security, eliminating the need to install DB2 logon exits in the DB2 systems to be accessed. RRSAF also opens up possibilities for new features in the Oracle Transparent Gateway for DB2.

If your installation was previously using RACF (or another security manager) to limit access to DB2 by DB2 session type (unlikely), you will need to change the security definitions to allow RRSAF access (instead of CAF).

DB2 Gateway-Specific DB2 Connect Exit Eliminated

With the switch to RRSAF, it is no longer necessary to modify the DB2 connect exit, DSN3@ATH.

If your installation previously modified Oracle's sample DB2 connect exit (from release 9.2 and earlier releases of the Oracle Transparent Gateway for DB2) and you want the same behavior in release 10*g*, you must make your own changes to the standard DB2 connect and DB2 sign-on exits (the sign-on exit is new for RRSAF). The DB2 sessions now come in as RRSAF instead of CAF types, so be sure to change

the exits to look for this new type of connection. The Oracle Transparent Gateway for DB2 uses RRSAF SIGNON to connect to the DB2 system. You will need this information if you decide to modify the standard DB2 connect or DB2 sign-on exits.

Typically an installation would run their DB2 system with the standard DB2 connect and DB2 sign-on exits supplied by IBM and these would need no modification to run release 10*g* of the Oracle Transparent Gateway for DB2.

DB2 Gateway Logon Exit

The Oracle Transparent Gateway for DB2 has a new logon security exit that verifies the user ID and password supplied when a session starts. It also sets up the security environment so that an `RRSAF SIGNON` has the necessary information for a user to sign on to DB2.

The old exit (prior to release 10*g*) was supplied only in source form and was compiled as part of the installation process. This exit was for CAF only, and is obsolete.

The new exit is shipped as an executable (`G4RRSAF`) in `AUTHLOAD`. Most installations should be able to use this exit with no changes. If it is required to have the source code or if your installation needs to make site-specific changes to the exit, then it is shipped in source form in the `SRCLIB PDS` but is called `G4SIGNON`, and the JCL to assemble and link the file is also in `SRCLIB`. If you decide to change the exit, be certain to link it with another name (suck as `G4SIGNON`), do not replace `G4RRSAF` in `AUTHLOAD`.

Most installations should be able to run with the DB2 Gateway logon exit (`G4RRSAF`) supplied in executable form in `AUTHLOAD`.

DB2 Date Exit

An optional DB2 date exit (`DSNXVDTX`) that recognizes the Oracle date formats was shipped only in source form and compiled and linked as part of the installation process prior to release 10.1.0.2.0

This exit is now shipped as an executable, `DSNXVDTX`, in the `AUTHLOAD`. It can be copied to the load libraries of DB2 systems if you want to use this exit. Source is supplied in `SRCLIB PDS` as member `DSNXVDTX` along with JCL to assemble and link it. The source is provided, in case your site requires the source for any exits installed in your DB2 systems. Typically, this exit would not be modified, but the source is available in case you need site-specific changes to this exit (or you need to combine this exit with your own DB2 date exit).

Most sites should be able to use the DSNXVDTX DB2 date exit supplied in AUTHLOAD. There is normally no need to make any changes to this (optional) DB2 date exit if you decide to install it in a DB2 system.

User Defined Function Support

DB2 user-defined functions (UDFs) can now be called directly from Oracle SQL. Refer to the *Oracle Database Heterogeneous Connectivity Administrator's Guide*, for details.

Oracle TIMESTAMP Data Type Mapping

Optionally, DB2 TIMESTAMP columns can be returned as CHAR(26) columns (as in previously releases) or mapped to Oracle's TIMESTAMP data type. This feature is activated by the ORACLE_TIMESTAMP environment variable.

Currently, if the Oracle TIMESTAMP data type is chosen, the microseconds are lost.

New DB2 Data Types Supported

DB2's data types ROWID, BLOB, CLOB, and DBCLOB have partial support in this release.

DB2 ROWID columns appear to Oracle as RAW(40) columns.

BLOB, CLOB, and DBCLOB data types (types of long objects, or LOBs) have only partial support in this release. Each of these column types appears as a RAW(4) to Oracle, and the value returned is the DB2 LOB locator. This may be usable if passed back to a client-written DB2 stored procedure. Currently, the data in any type of LOB cannot be returned. This may be implemented in a future release.

OSDI IDLE_TIMEOUT

The new OSDI IDLE_TIMEOUT service parameter can be used to time out and automatically cancel any Oracle Transparent Gateway for DB2 session that has been idle for a specified amount of time.

Changes and Enhancements in Release 10.2.0.2.0

Full UNICODE Support

The DB2 Gateway now runs internally in UNICODE. All DB2 tables, EBCDIC, ASCII and UNICODE, using any DB2 character set, are supported.

Because the gateway now runs internally in UNICODE, HS_LANGUAGE can no longer be specified. HS_LANGUAGE is internally set to AL32UTF8 and cannot be changed.

Any DB2 table in any character set can now be accessed or updated.

The controlling factor in character set conversion now becomes the database character set of the Oracle integrating server. Ideally, the database character set of Oracle should be AL32UTF8. If you have an Oracle running in UTF8, you should convert it to AL32UTF8 (UTF8 is a subset of AL32UTF8). If you have an Oracle running in US7ASCII(the default Oracle database character set on ASCII platforms), you should convert it to AL32UTF8 (US7ASCII is also a subset of AL32UTF8).

All data retrieved from DB2 is converted from the DB2 character set directly to the database character set of the Oracle integrating server. If the Oracle server is running in AL32UTF8, all possible code points will be converted. If the Oracle server is running in any other character set (like WE8ISO8859P1), some code points may not have an equivalent and will be converted to the substitution character for that character set.

All SQL statements and input bind variable values for SQL statements will be provided to DB2 in UNICODE. This allows any code point in any character set to be sent to DB2. If the Oracle integrating server isn't running in AL32UTF8, then the SQL statement and bind variables will be converted from the Oracle server's character set to AL32UTF8. For this reason, it is highly recommended to run the Oracle server in AL32UTF8.

All passthrough SQL statements will be sent to DB2 in UNICODE. If there are differences in how a SQL statement must be coded for UNICODE (like for DB2 GRAPHIC literals), the UNICODE form must be used.

This new UNICODE feature allows for the maximum accessibility to DB2 data. Every code point in every DB2 character set can now be retrieved from DB2 or provided as input to DB2. The only limiting factor is the database character set of the Oracle integrating server.

LOB Support

BLOB, CLOB and DBCLOB data types (types of long objects, or LOBs) are supported in this release. Where only a RAW(4) LOB locator was returned previously (see new features for 10.1.0.2.0 above), in this release, the actual data values of the LOB are returned. BLOB datatypes appear as LONG RAW, and, CLOB and DBCLOB datatypes appear as LONG types. The entire value is retrieved on a SELECT, just like a regular LONG RAW or LONG column.

SAVEPOINT and ROLLBACK TO SAVEPOINT Support

SAVEPOINT and ROLLBACK TO SAVEPOINT are now supported.

Improved Data Dictionary Translations

The Data Dictionary Translations, which simulate several Oracle tables (like ALL_TABLES and ALL_TAB_COLUMNS) retrieving data from the DB2 catalog have been significantly enhanced. The improvements provide a closer simulation to the Oracle equivalent data dictionary tables. The performance and accuracy of the translations has also been significantly improved.

Support is now provided for the DB2 7.1 and DB2 8.1 catalogs (which are different). The longer names in DB2 8.1 are now returned in the data dictionary translations and some new values available in DB2 8.1 are returned as well.

External security data dictionary translations are also provided.

If you use DB2's external security (an exit is called which typically makes RACF calls to check for DB2 object security instead of DB2 checking for GRANTs), then new views are available for you to use.

The problem with DB2's external security is that the GRANT information in the DB2 catalog is not used. There are no DB2 tables which contain GRANT information, all of the access information is in an external security system (typically RACF). Consequently, the data dictionary translations cannot check the DB2 catalog to see if a user has access to an object.

The alternative external security views simply return all objects rather than just the objects a user has access to. For example, the standard security view for ALL_TABLES returns just the tables that a user has been granted access to through DB2 GRANT commands. However, the external security views simply return every table in the DB2 catalog for ALL_TABLES (since there is no grant information in the DB2 catalog to check against).

With the external security views, a user will be able to retrieve the definition of all of the objects in the DB2 catalog. However, the user will only be able to access the objects that the external security exit allows access to.

Oracle Timestamp Data Type Mapping

Mapping of DB2 TIMESTAMP columns to Oracle's TIMESTAMP data type was introduced in 10.1.0.2.0 (see new features for that release above).

Previously, the DB2 TIMESTAMP's microsecond's were lost. In this release, the full timestamp, including the microseconds, are included in the Oracle TIMESTAMP value.

DB2 Gateway Logon Exit Enhanced

Several enhancements were made to the DB2 Gateway logon exit. The SAF call was enhanced to improve performance of the security checking call and a change was made to suppress the console log message that appeared every time the security check was done.

NCHAR Support Dropped

The `NCHAR` data type is no longer supported. All DB2 `GRAPHIC`, `VARGRAPHIC` and `DBCLOB` datatypes are converted to their `CHAR` equivalents (`CHAR`, `VARCHAR` and `LONG` respectively).

Known Problems

There are no known problems in this release of the gateway.

Known Restrictions

The restrictions documented in this section are known to exist for the products included in the installation media. Also refer to Chapter 9, "Developing Applications" for information about limitations when developing applications.

Owners of DB2 Components

DD Basic Tables and Views The owner of DD basic tables and views is OTGDB2. This cannot be changed.

Owner of DB2 Plan The qualifier for the DB2 plan must be the same as the owner for the `ORACLE2PC` table in the DD SQL script, and it must be `OTGDB2`.

Binary Literal Notation

Oracle SQL uses hexadecimal digits surrounded by single quotes to express literal values being compared or inserted into columns defined as `RAW`. Currently, this is not converted to DB2 syntax (an X followed by quoted hexadecimal digits) when the SQL destination is the gateway. You must use bind variables to compare or insert into a DB2 server column defined with the `FOR BIT DATA` option.

Programmatic Limitations

Gateway design requires that all host variables in a SQL operation be bound before performing a describe function. When using the Oracle Call Interface (OCI), all OCI

bind calls for a given statement must be completed before an OCI describe call is made.

Columns Defined with RAW Data

When you select RAW data into character bind variables, the CHAR column must be two times the size of the RAW data. Selecting RAW data into character bind variables causes implicit RAW to HEX conversion. If the character bind variable column is too small, then the SELECT statement fails.

Precompiler Limitations

The SQLCHECK option must be set to NONE when precompiling programs with the Oracle Precompilers.

SQL Limitations

Although most differences between the Oracle database server SQL and DB2 SQL are handled by the gateway, the following restrictions exist:

- Oracle ROWID is not supported.

  DB2 does not have a functional equivalent to Oracle ROWID. Tools or applications depending on Oracle ROWID are not supported.

- UPDATE and DELETE with the WHERE CURRENT OF CURSOR clause are not supported.

  When these statements are used in precompiler and PL/SQL programs, they rely internally on the Oracle ROWID function. Therefore, they are not supported.

- Oracle bind variables become DB2 parameter markers.

  Oracle bind variables become DB2 parameter markers when used with the gateway. Therefore, the bind variables are subject to the same restrictions as DB2 parameter markers. For example, the following statements are not allowed:

  ```
  WHERE :x IS NULL
  WHERE :x = :y
  ```

  For more information about DB2 parameter marker restrictions, refer to the IBM documents for your platform and operating system.

- CONNECT BY is not supported.

  The Oracle-specific CONNECT BY clause is not supported.

GTWY_IDLE_TIME

`GTWY_IDLE_TIME` under the previous MPM architecture was replaced by OSDI `IDLE_TIMEOUT`. See Section , "IDLE_TIMEOUT | ITIMEOUT" on page 5-10.

3

System Requirements

This chapter lists the hardware and software requirements for installing the gateway. It includes the following sections:

Resource Requirements

The following Oracle Transparent Gateway for DB2 resource requirements are discussed in this chapter:

- CPU
- Disk space
- Virtual memory

CPU

The gateway requires any processor that can run the required z/OS operating systems listed in "Software Requirements" on page 3-2.

Disk Space

The following are the disk space requirements:

- Temp z/fs: 600 MB.

  If you want to further reduce the requirement of temporary z/fs disk space, then delete the zip file after uncompressing it, and before running the installer. After running the Oracle installer, if you are sure you would not need to configure another gateway instance, then you can delete the installer home which is the result of uncompressing the zip file.
- Permanent z/fs: 220 MB with locale builder, 166 MB without locale builder
- Permanent Data Set Space: 200 cylinders

Virtual Memory

Each concurrent user of the gateway increases the virtual memory consumption, and it varies depending on:

- The number and complexity of SQL statements referring to tables through the gateway
- The number of columns and column sizes of those tables
- The configured network buffer size
- The number of open cursors and the `HS_RPC_FETCH_SIZE` initialization parameter

Oracle recommends defining `0M` for the `REGION` parameter in the Oracle Transparent Gateway for DB2 address space.

Oracle also recommends defining `0M` for the `REGION` parameter in the Oracle Net address space.

Software Requirements

Oracle Transparent Gateway for DB2 has requirements for the following:

- Operating System
- UNIX System Services (USS)
- DB2 Maintenance

The system software configuration described in the following requirements is supported by Oracle, as long as the underlying system software products are supported by their respective vendors. Verify the latest support status with your system software vendors.

Operating System

z/OS 1.4 (or V1R4) or higher is required.

UNIX System Services (USS)

An operational z/OS UNIX System Services (USS) is required.

IBM Maintenance

Oracle recommends running Oracle Transparent Gateway for DB2 at the most current operating system service level.

DB2 Maintenance

If you are using the current version of the Oracle Transparent Gateway for DB2 to access DB2 v7.1 you need to apply the following DB2 maintenance:

`APAR PK20664: SQLCODE330 ON DESCRIBE OF A PREPARED STATEMENT USING BIND OPTION ENCODING(UNICODE).` The corresponding `PTF` is `UK13641.`

Oracle Database Server Requirements

The Oracle Server which is to act as the Oracle integrating server requires the latest patch set for Oracle Database 10*g* and Oracle9*i* release 9.2.

Oracle Net Requirements

When you install Oracle Net, you have additional software and hardware requirements.

Oracle Net Protocol	Product	Required Protocol Release
Oracle Net IBM TCP/IP	z/OS	1.4 or higher

Oracle Net IBM TCP/IP

A z/OS host using Oracle Net IBM TCP/IP requires hardware capable of interfacing with the IBM TCP/IP stack.

The IBM TCP/IP software must be on the same z/OS system as Oracle Net IBM TCP/IP.

Distribution Kit

Before installing the gateway, verify that you have the correct installation media and proper documentation:

- Oracle Database product installation media
- Oracle documentation for installing, administering, and using the gateway

4

Installation

This chapter describes installing the Oracle Transparent Gateway for DB2. It includes the following sections:

Each of these steps is described in this chapter. Installation messages are documented in the *Oracle Database Messages Guide for IBM z/OS (OS/390)*.

To configure a gateway service, refer to Chapter 5, "Configuring a Gateway Service".

To create the Net service, refer to Chapter 6, "Oracle Net".

For information about migration, refer to Chapter 11, "Migration and Coexistence with Existing Gateways".

Installation Checklists

Use the following checklists for installation.

Installation Checklist

Post-installation Checklist

Product Installation

Perform the following steps to install the Oracle Transparent Gateway for DB2:

Step 1: Perform Preinstallation Tasks

Oracle Transparent Gateway for DB2 is part of the Oracle Database for z/OS distribution. Before you can install the software, you need to perform various preinstallation tasks. For preinstallation instructions, refer to the *Oracle Database Installation Guide for IBM z/OS (OS/390)*. After you have performed the required preinstallation tasks, you will be referred back to this manual to perform installation tasks specific to the Oracle Transparent Gateway for DB2. At that point, continue with the following steps.

Step 2: Install the Oracle Transparent Gateway for DB2 Software

The following sections describe how to install the Transparent Gateway for DB2 10*g* for z/OS Software:

Review Product-Specific Installation Guidelines

Review the following guidelines before starting the Installer:

- Using the Oracle Universal Installer

Use the installer from the current release only to install components from this release. An installer from an earlier release will not install the necessary components.

- Reinstalling the Gateway Software

  If you reinstall gateway software into an Oracle home directory where a gateway is already installed, you must also re-install any components that were installed before you began the re-installation.

Run the Installer and Install the Software

Run the Installer and install the software, as follows:

1. Log in as the Oracle software owner user (the user performing the installation) and set the `DISPLAY` environment variable, if necessary.
2. To start the Installer, enter the following commands, where *`directory_path`* is the path of the `Disk1` directory on the hard disk:

   ```
   $ /directory_path/runInstaller
   ```

   If the Installer does not appear, refer to the Troubleshooting appendix of the *Oracle Database Installation Guide for IBM z/OS (OS/390)* for information on troubleshooting.
3. Use the following guidelines to complete the installation:
   - Follow the instructions displayed in the Installer windows. If you need additional information, click **Help**.
   - When the Installer prompts you to run a script with root privileges, enter a command similar to the following in a terminal where you are logged in as the root user, then click **Continue** or **OK**:

     ```
     $ /oracle/v10g/root.sh
     ```

   - If you encounter errors while installing or linking the software, refer to the Troubleshooting appendix of the *Oracle Database Installation Guide for IBM z/OS (OS/390) z/OS* for information on troubleshooting.
4. On the **Select a Product to Install** panel, there are two options. Select the option **Oracle Transparent Gateway for DB2 10.2.0.2.0**, to install the gateway software. You must install the software before configuring it.

You can configure the gateway within the same Installer session or at a later time by running the Installer again and choosing the **ConfigureOracle Transparent Gateway for DB2 10.2.0.2.0** option.

Run the Generated Installation Job

1. Run the `CPLNKLST` job in the `INSTLIB` library of the installation High Level Qualifier (HLQ) to move the load modules from `AUTHLOAD` to the linklist library. Note that if you choose different HLQs for the installation and configuration steps two `INSTLIB`s will be generated. This is the only step referring to installation-HLQ `INSTLIB`. For the rest of the book, `INSTLIB` refers to the configuration-HLQ `INSTLIB`.
2. Refresh the linklist by issuing the following command to a z/OS console:

```
F LLA,REFRESH
```

Postinstallation Steps

After installing the Oracle Transparent Gateway for DB2 software, you must perform the following post-installation steps. These changes must be made and the z/OS system IPLed before you can configure a gateway service.

Step 1: Download and Install Patches

Step 2: Add Program Properties

Step 3: Authorize the Gateway Load Library

Step 4: Re-IPL z/OS or Use Dynamic z/OS Commands

Step 1: Download and Install Patches

Check the Oracle*Metalink* Web site for required patches for your installation. To download the required patches, perform the following steps:

1. Using a Web browser, go to the following Web site:

```
http://metalink.oracle.com
```

2. Log in to Oracle*Metalink*. If you are not an Oracle*Metalink* registered user, then click **Register for Metalink!** and follow the registration instructions.
3. On the main Oracle*Metalink* page, click **Patches**.
4. Use the Simple Search feature to search for Product or Family. Then, specify the following information:

- In the Product or Family field, specify the option for the Oracle Transparent Gateway for DB2.
- In the Release field, specify the current release number.
- In the Patch Type field, specify **Patchset/Minipack**.

When you have filled in the required information, click **Go**.

Step 2: Add Program Properties

The gateway and network service region programs must run nonswappable and noncancelable, and should not be subject to system time limits. In addition, the gateway service runs in protect key 7. These attributes are indicated by adding entries for these programs to the z/OS Program Properties Table (PPT), through a member of the `SYS1.PARMLIB` data set named `SCHEDxx`, where *xx* is a 2-letter or 2-digit suffix. You may need to work with your systems programmer to determine the correct member name and to add the entries. The entries that you add should be similar to those in the following example. The comments, which are included for clarity, are allowed but are not required.

```
  /* SCHEDxx PPT entry for Oracle gateway region */
PPT PGMNAME(ORARASC)    /* Program (module) name  */
  NOCANCEL              /* Not cancelable         */
  KEY(7)                /* Protection key         */
  NOSWAP                /* Not Swappable          */
  SYST                  /* Not subject to timing  */
  /* SCHEDxx PPT entry for Oracle network region  */
PPT PGMNAME(ORANET)     /* Program (module) name  */
  NOCANCEL              /* Not cancelable         */
  NOSWAP                /* Not swapable           */
  SYST                  /* Not subject to timing  */
```

The entries in the `SCHEDxx` member are normally read at z/OS IPL. You can cause z/OS to re-read the member without an IPL by using the `SET SCH` operator command. The PPT entries must take effect before Oracle gateway and network services are started.

For details about the `SCHEDxx` member, the PPT, and the `SET SCH` command, refer to the IBM manuals *MVS Initialization and Tuning Reference* and *MVS Systems Commands*.

Step 3: Authorize the Gateway Load Library

You must APF-authorize the gateway `AUTHLOAD` library.

To perform this step, first select your gateway data set name qualifiers. If an Oracle Database 10*g* server is currently installed, then these qualifiers must be different from the qualifiers used for the Oracle Enterprise for OS/390 server. The disk volume for the gateway AUTHLOAD library is also specified in this step.

Because future installation procedures for updates or service might require Oracle modules to reside in their own authorized library, Oracle Corporation does not recommend installing the authorized Oracle load modules in an existing authorized library.

To mark the gateway load library as authorized, use ISPF EDIT, the IEBUPDTE utility, or an equivalent function to add the gateway load library name to the appropriate PROGxx or IEAAPFxx member in SYS1.PARMLIB (where xx is the value in the APF=xx parameter specified in IEASYSnn or by the system operator when the z/OS system is IPLed).

You can also authorize the load library in the JCL for the gateway as you proceed through the installation. For example:

```
// SETPROG APF,ADD,DSN=name_of_your_load_library,VOL=VOLSER
```

This is generally the simplest method of APF authorizing a library. For SMS-managed libraries, specify SMS instead of VOL=VOLSER.

Step 4: Re-IPL z/OS or Use Dynamic z/OS Commands

The Oracle AUTHLOAD library can be APF-authorized dynamically by using the SETPROG system command. You can use these commands to avoid an IPL. However, you still need to make the changes to the SYS1.PARMLIB members as discussed in "Step 3: Authorize the Gateway Load Library". These changes to the SYS1.PARMLIB members are necessary for APF entries to be permanent.

You must IPL z/OS or use the SETPROG system commands to ensure the Oracle AUTHLOAD library is APF-authorized before you bring up a gateway.

Summary

After you have completed the pre-installation, installation, and post-installation tasks described in this chapter, the Oracle Transparent Gateway for DB2 software installation is complete. These tasks are performed only once.

The MVS data sets, HFS files, and environment you created for the installation are required for configuring a gateway service and for applying patches with OPATCH (which is how the patches for the DB2 gateway will be distributed from now on). Be

sure that none of these files are deleted. To configure a gateway service, refer to Chapter 5, "Configuring a Gateway Service". This may be done more than once.

To create the Net service, refer to Chapter 6, "Oracle Net".

For information about migration, refer to Chapter 11, "Migration and Coexistence with Existing Gateways".

5

Configuring a Gateway Service

After you have created an Operating System Dependent Interface (OSDI) subsystem, you can configure one or more gateways to run under that subsystem. This chapter describes how to set up OSDI definitions, JCL procedures, parameter files, and other z/OS-specific items required by a gateway instance. If you are new to OSDI, read this chapter to learn how OSDI differs from the MPM subsystem as far as gateway configuration is concerned.

This chapter includes the following sections:

Overview

To create a gateway instance under OSDI, you must first define the instance as a service using the `OSDI DEFINE SERVICE` command. In addition to defining the service, some other items must be set up before the service can be started: a JCL procedure, several parameter files, and possibly security resource definitions.

After you have defined the instance as a service and set up the additional items, you can start the service, which creates one or more z/OS address spaces based on

controls that you have specified. A description of the configuration process is included in this chapter on page 5-24.

Gateway Service Definition

The `OSDI DEFINE SERVICE` command is described completely in Appendix A, "OSDI Subsystem Command Reference". Here, we cover `DEFINE` parameter considerations that are specific to a gateway service.

Service Name

The service name for a gateway can be anything that you want within the content limitations described in Appendix A. By default, OSDI will use the service name as the SID for the service. (The SID is an identifier that users or application developers must supply to connect an application to a particular database.) The SID can be specified separately, however, and is not required to be the same as the service name.

Note: If you specify a service name that is the same as any existing subsystem name in your system (gateway or otherwise), then you must also specify a `JOBNAME` parameter that is not the same as any existing subsystem. If you do not use unique names, then OSDI starts the service using the service name as the job identifier. When z/OS processes a start for an address space whose job name or job identifier matches a known subsystem, the job runs under control of the master subsystem instead of under control of JES.

Caution: Running OSDI services under the master subsystem is not supported. This situation must be avoided by making sure that the service runs with a job name or a job identifier that is not the same as any subsystem name.

TYPE

The `TYPE` parameter for a gateway service must be specified as `GTW`.

PROC

This parameter specifies the name of a service JCL procedure that you will place in one of your system procedure libraries. The procedure need not exist when DEFINE SERVICE is issued, but it must be in place before the service is started. The procedure name can be anything that you choose or that the naming standards of your installation require. The requirements for this procedure are discussed in section "Gateway Region JCL" on page 5-4.

PARM

The PARM parameter for a gateway service specifies the name of a z/OS data set containing service initialization parameters. These are z/OS-specific parameters and are described in the section "Gateway Region Parameters" on page 5-7. Typically, PARM will specify a member of a Partitioned Data Set (PDS) that is used for various Oracle parameter files. If no member name is included in the PARM string, then the specified data set must be sequential (DSORG=PS).

MAXAS

If you want to exploit the multiple-address-space server features of OSDI, then you should specify the MAXAS parameter on DEFINE SERVICE with a value greater than the default of 1. This sets the maximum number of address spaces for the service, which may be greater than the number started when the service is first brought up. (The number of address spaces to start initially is a gateway region parameter.) This parameter can be altered with OSDI commands as long as the gateway service is not active.

JOBNAME

When you run a gateway service with multiple address spaces, the JOBNAME parameter of DEFINE SERVICE can be used to cause each address space to have a distinct jobname. Although this is not required, it may be desirable if you use z/OS facilities (such as RMF) that distinguish address spaces by jobname. To do this, specify JOBNAME *(name*)*, where *name* is a one-character to five-character jobname prefix followed by an asterisk, as shown. As each address space is started, OSDI substitutes a three-digit address space counter for the asterisk (001, 002, and so on) to produce the final jobname. You can also use JOBNAME to cause the service to run with a jobname different from the service name (which is used by default).

As discussed in the "Service Name" section, you must specify a JOBNAME parameter if the service name matches any existing subsystem name in your z/OS system.

SID

The SID parameter specifies a unique identifier for the service. It is a critical element in the process that is used by Oracle database applications to specify the instance to which they must connect. (Inbound network clients specify a SID parameter in the Oracle database network address string that must match the SID that is specified in DEFINE SERVICE).

Although you can run the OSDI DEFINE SERVICE command through a z/OS system console or similar facility, you should put definition commands for services that you use regularly into the OSDI subsystem parameter file, after the DEFINE SERVICEGROUP command. This ensures that the service is always defined correctly and automatically when the subsystem is initialized (normally at system IPL). In the following sample gateway DEFINE SERVICE command, the command prefix has been omitted and continuation hyphens have been included as though the command were in the subsystem parameter file:

```
DEFINE SERVICE DB2GW1 TYPE(GTW) PROC(DB2GW1) -
 DESC('Test Gateway') -
 SID(GTW1) PARM('ORACLE.GTW1.PARMLIB(DB2G1P)')
```

Gateway Region JCL

Defining a gateway service requires you to specify a JCL procedure name in a system procedure library. You must create the procedure before you try to start the service, and the procedure must invoke the OSDI gateway region program with an EXEC statement such as the following:

```
// EXEC PGM=ORARASC,REGION=0M
```

REGION=0M is specified to ensure that the server can allocate as much private virtual memory as it needs. Some z/OS systems may prohibit or alter a REGION parameter such as this, so you might want to check with your systems programmer to make sure that the system will accept your REGION parameter.

A z/OS exit called IEFUSI might be installed on your system. The IEFUSI exit prevents started tasks or batch jobs from getting the maximum region size when REGION=0M is specified. If an IEFUSI exit is implemented, then it is specified in the SMFPRMxx member of SYS1.PARMLIB that is used during z/OS initialization. To effectively run the Oracle database with an IEFUSI exit installed, ensure that the exit is coded to allow batch jobs or started tasks with the names of your Oracle regions to allocate a large amount of virtual memory above the 16M line.

Because the gateway allocates only the amount of memory it needs, you can safely allow it to allocate any amount of memory up to the two-gigabyte limit per address space which is imposed by 31-bit addressing conventions.

Note that no other EXEC statement parameters are needed. The PARM parameter of EXEC is not used by the database region program.

Changing the JCL procedure after starting one or more address spaces for the service, and then starting another address space (to use the changed JCL), is not supported.

In addition to the EXEC statement, the procedure will need several DD statements, as follows:

ORA$ENV

This DD statement is optional. When used, it specifies a sequential file or PDS member containing environment variable assignment statements. Environment variables are used to supply operating parameters to certain gateway product features. Reliance on environment variables and considerations for setting them are discussed in feature-specific chapters of this manual. The data specified by ORA$ENV is read-only at gateway service startup. Therefore, in order for changes to the data set to take effect, the service must be stopped and started.

Be aware that the global environment variable file is not read by the gateway. All environment variable settings for the server must be supplied through ORA$ENV. For more information on the global environment variable file, refer to the *Oracle Database System Administration Guide for IBM z/OS (OS/390)*.

ORA$FPS

This DD statement specifies a sequential file or PDS member containing z/OS-specific parameters that control data set processing in the gateway. These parameters are organized by type of file, and they primarily pertain to creation processing when the gateway uses dynamic allocation to create a z/OS trace data set. For considerations and syntax rules for the ORA$FPS parameter file, refer to the *Oracle Database System Administration Guide for IBM z/OS (OS/390)*.

The ORA$FPS DD is optional. If you omit it, then file creation operations may fail unless your installation has DF/SMS ACS routines that supply defaults for data set creation parameters. At gateway service startup, data specified by ORA$FPS is read and checked. Any errors are reported and ignored. Valid entries are loaded as server file management parameters. After gateway service startup, a new set of server file management parameters can be loaded from the updated ORA$FPS

specification by using the REFRESH FPS command. For more information, refer to the *Oracle Database System Administration Guide for IBM z/OS (OS/390)*.

> **Note:** When this DD statement is omitted, an IEC130I message may appear in the system log during service address space initialization. This is normal.

ORA$LIB

This DD statement specifies a non-authorized load library from which non-executable (data) modules are fetched. The modules contain Globalization Support data objects and messages that are associated with Oracle Globalization Support internationalization features. Normally these modules are installed in the OSDI MESG data set, for example ORACLE.V10G.MESG. The ORA$LIB DD statement is optional: if you omit it, then the Oracle server attempts to fetch messages and Globalization Support data objects modules from STEPLIB. Do not concatenate a non-APF-authorized MESG data set to STEPLIB in lieu of specifying ORA$LIB.

> **Note:** When this DD statement is omitted, an IEC130I message may appear in the system log during service address space initialization. This is normal.

SQLNET

This DD statement specifies an input file containing Oracle Net parameters. It is required if the Oracle instance uses any of the following:

- Network data encryption
- Network activity tracing
- Altering of default Oracle Net file names

Refer to Chapter 6, "Oracle Net", for additional information.

STEPLIB

This DD statement must specify the APF-authorized Oracle AUTHLOAD library that was populated during installation. The IBM LE/370 run-time library must be concatenated to it unless your installation has put the LE/370 run-time into the system linklist. A typical name for the LE/370 run-time library is SYS1.SCEERUN,

but it may have a different name in your system. The DB2 load library (SDSNLOAD) must be specified unless it is in the linklist.

SYSPRINT

This DD statement is optional. When used, the gateway instance alert log is written to it. Regardless of the number of server address spaces, an Oracle database instance has only one alert log, which is opened by the first server address space (AS1). Alerts that are generated by sessions in other address spaces are routed to AS1.

You can specify a sequential (DSORG=PS) disk data set or a spool file (SYSOUT) for this DD. If you omit the SYSPRINT DD, then the alert log is dynamically allocated as a disk data set or spool file according to the ALERT_DSNAME region parameter, discussed in "Gateway Region Parameters" on page 5-7.

If you specify a disk data set for SYSPRINT and an error occurs while it is being written (including an out-of-space condition), an alert log switch occurs. Refer to the *Oracle Database System Administration Guide for IBM z/OS (OS/390)* for additional information on Oracle alert log switching.

Sample Gateway Region JCL Procedure

The following is an example of a JCL procedure for a gateway region:

```
//DB2GW1 PROC
//ORACLE EXEC PGM=ORARASC,REGION=0M
//STEPLIB  DD  DISP=SHR,DSN=ORACLE.V10G.AUTHLOAD
//         DD DISP=SHR, DSN=DSN710.SDSNLOAD
//         DD  DISP=SHR,DSN=SYS1.SCEERUN
//ORA$LIB  DD  DISP=SHR,DSN=ORACLE.V10G.MESG
//ORA$FPS  DD DISP=SHR,DSN=ORACLE.ORA1.PARMLIB(G4DB2FPS)
//ORA$ENV  DD DISP=SHR,DSN=ORACLE.ORA1.PARMLIB(G4DB2ENV)
//SQLNETLG DD SYSOUT=*
```

Gateway Region Parameters

OSDI gateway region parameters are supplied in a data set whose name is specified as the PARM string in the service definition. This will typically be a member of a PDS. Because the data set name is supplied through the service PARM mechanism, no DD statement is coded in the region JCL. The data set is dynamically allocated, opened, and read when the service is started. Changing parameters in the data set has no effect until the service is stopped and restarted.

Region parameters are read independently by each address space of a multi-address space server. Adding, removing, or changing parameters between the starting of one address space and later starting of another is not supported.

The OSDI gateway region parameters consist of a parameter name followed by the parameter value in parentheses. Each parameter has a long descriptive name and a shorter name of eight characters or less. Each record may contain only one parameter. No continuation is allowed. Records beginning with an asterisk (*) are treated as comments and are ignored. Embedded spaces and all characters after the closing parenthesis are ignored.

ALERT_DSNAME | ADSN

ALERT_DSNAME specifies a filespec value for the gateway alert log for the purpose of alert log switching. The format for this parameter is as follows:

```
ALERT_DSNAME ( filespec )
```

The *filespec* value can be a SYSOUT type or a data set name type with embedded system symbols that will guarantee a unique data set name on each use. Using date and time system symbols is recommended in a data set name filespec value.

Examples:

```
ALERT_DSNAME(ORACLE.&ORASRVN..ALERT.D&LDATE..T&LTIME)
ADSN(//S:Z,,DBOPS01)
```

If you omit this parameter, then alert log switches use a default SYSOUT specification //SYSOUT:*.

ALERT_MAX | AMAX

ALERT_MAX specifies:

```
ALERT_MAX ( size )
```

The *size* value is the number of data bytes (sum of logical record lengths) written to the alert log. This value can be specified as a number, *n*, *n*K (denoting a multiplier of 1024), or *n*M (denoting a multiplier of 1,048,576). The writing of an alert log record that would exceed this size causes an automatic alert log switch before the new record is written. This happens without regard for the sequence or interrelationship between alert log messages, for example, the switch can occur between a pair of related messages.

The default value is 0. If you specify 0, then no automatic switching is done. Specifying a value less than 65536 (64 K) is not recommended for this parameter.

ALERT_MIN | AMIN

ALERT_MIN specifies:

```
ALERT_MIN ( size )
```

The *size* value is the number of data bytes (sum of logical record lengths) written to the alert log. This value can be specified as *n*, *n*K (denoting a multiplier of 1024), or *n*M (denoting a multiplier of 1,048,576). If an alert log switch is requested (for example, using a LOGSWITCH service command without the FORCE option), then the request is honored only if the size of the current alert log exceeds this value.

The default value is 0. If you specify 0,then no minimum size checking is done, and all alert log switch requests are carried out.

DSN_PREFIX_DB | ORAPREFD

The DSN_PREFIX_DB parameter supplies a constant string that is associated with the &ORAPREFD system symbol. The &ORAPREFD system symbol can be used to form the high-level (leftmost) qualifier of z/OS data set names generated by the gateway. The format is as follows:

```
DSN_PREFIX_DB ( dsn_prefix )
```

The dsn_prefix value is a valid one-character to eight-character data set name qualifier that conforms to the installation requirements. In most cases, this will be the qualifier that is used for all Oracle database files associated with this instance. For example:

```
DSN_PREFIX_DB(ORADB01)
```

DSN_PREFIX_DB has no default value. If you omit this parameter, then certain situations in which the gateway generates default file names will produce errors.

DEDICATED_TCB

DEDICATED_TCB specifies whether OSDI would assign a dedicated z/OS TCB for each session. For gateways running on z/OS, it must be set to AUTO. This parameter and its value are automatically filled in upon the installation of the gateway. The format is:

```
DEDICATED_TCB(AUTO)
```

IDLE_TIMEOUT | ITIMEOUT

The `IDLE_TIMEOUT` parameter sets a timeout value for idle sessions. Sessions that are idle for a period longer than the interval set are terminated, and all resources are released. The format is as follows:

```
IDLE_TIMEOUT ( time_interval )
```

The *time_interval* value is the timeout value specified as *nnn* or *nnn*S for seconds or *nnn*M for minutes. There is no default timeout value. The maximum value is 604,800 seconds or one week. The timeout value set is a minimum approximation, and a session may be idle for some additional seconds or minutes before it is terminated. When a session is using a dedicated TCB, as is the case with Transparent Gateway products, the task is terminated with an S222 completion code. Clients with connections to timed-out sessions may see a variety of errors if they attempt to continue.

INIT_ADR_SPACES | INTADSPC

`INIT_ADR_SPACES` controls how many auxiliary address spaces are started. The format is as follows:

```
INIT_ADR_SPACES ( number_of_address_spaces )
```

The *number_of_address_spaces* value is the number of address spaces to start. The default is 1, which starts only the control address space (AS1). The maximum is the number that was specified for `MAXAS` on the associated `DEFINE SERVICE` command for the gateway service.

INIT_STACK_SIZE | INTSTKSZ

`INIT_STACK_SIZE` controls the size of the C stack that is allocated for each session. The format is as follows:

```
INIT_STACK_SIZE ( init_size )
```

The *init_size* value determines the initial size of the C stack. This value can be specified as *n* or *n*K. The default is 128 K. This is the size recommended for the gateway.

LOGON_AUTH | LGNAUTH

LOGON_AUTH specifies how the gateway interacts with a SAF-based external security product when processing logons. The format is as follows:

```
LOGON_AUTH ( auth )
```

You can specify the *auth* value by using one of the parameters in the following table.

Parameter	Usage
G4RRSAF	call standard logon exit for gateway
exitname	call an installation-supplied logon exit; *exitname* is the one-character to eight-character load module name of the exit

If *exitname* is specified, then it must reside in the system linklist or in an APF-authorized library that is part of the server region STEPLIB concatenation. The default is NONE.

Examples:

```
LOGON_AUTH(G4RRSAF)
```

For more information about gateway logon authorization, refer to "Gateway Security" on page 7-4.

MAX_SESSION_MEM | MAXSMEM

The MAX_SESSION_MEM parameter specifies a hard limit on the amount of virtual memory that a single gateway session can allocate. The format is as follows:

```
MAX_SESSION_MEM ( session_memory )
```

The *session_memory* value is the maximum amount of virtual memory that a single gateway session can allocate. This value can be specified as *n*, *n*K (denoting a multiplier of 1024), or *n*M (denoting a multiplier of 1,048,576). The default is zero (0), which means no session limit is imposed.

This parameter is useful for stopping a runaway session that is allocating excessive amounts of memory due, perhaps, to problems with application design. This pertains only to session-private C stack and heap memory allocated during gateway processing. It does not include internal memory allocations done by the implementation (for example, DB2).

MAX_SESSIONS | MAXSESS

The MAX_SESSIONS parameter limits the number of sessions that can be scheduled in an address space. The format is as follows:

```
MAX_SESSIONS ( number_of_sessions )
```

The *number_of_sessions* value is the maximum number of sessions per address space. This value can be specified as *n* or *n*K. The default is 1024. The number of sessions that can be supported in an address space depends on the complexity of the work. Limiting the number of sessions per address space reduces the chances of session failure due to exhaustion of virtual storage.

PRIMARY_ASC_MODE | PRIMASCM

The PRIMARY_ASC_MODE parameter specifies whether the gateway or the Oracle server being executed will be given control in the PRIMARY Address Space Control (ASC) Mode. For the Oracle Transparent Gateway for DB2, PRIMARY_ASC_MODE(YES) must be specified.

PRIMARY_ASC_MODE(NO) is the default, if the parameter is not specified. The default, or the use of NO, specifies that the Gateway or the Oracle server is to be given control in Access Register (AR) mode. The Oracle Transparent Gateway for DB2 can run only in PRIMARY Address Space Control Mode, whereas the Oracle server can run in AR mode without a problem.

REGION_MEM_RESERVE | REGMRES

The REGION_MEM_RESERVE parameter specifies the amount of private area memory in the server address space to be reserved for implementation and z/OS use. The format is as follows:

REGION_MEM_RESERVE (region_memory)

The region_memory value is the amount of private area memory reserved. This value can be specified as *n*, *n*K (denoting a multiplier of 1024), or *n*M (denoting a multiplier of 1,048,576).

During initialization, each server address space calculates the total available private area memory and subtracts the reserve amount from it. The result is the aggregate limit for all session memory requests in that address space.

The default is zero (0), which means that no aggregate limit applies. In this case, it is possible for session memory requests to exhaust the available private area of the address space, leading to unpredictable failures.

Thus, the reserve amount must be sufficient to accommodate internal implementation memory requirements as well as memory required by z/OS services used by the gateway, particularly Local System Queue Area (LSQA) memory. Because it is difficult to predict this amount for any given workload, the best strategy is to specify a relatively large reserve amount, such as 50 M or more. This has the effect of reducing slightly the number of sessions that can be accommodated in a gateway address space. However, additional address spaces can be started, if necessary.

SERVER_LOADMOD | SRVRLMOD

SERVER_LOADMOD specifies the name of the service load module. The format is as follows:

```
SERVER_LOADMOD ( loadmod )
```

The *loadmod* value is the name of the load module to load. For the gateway, this is usually G4DB2SRV. This parameter is required.

SMF_STAT_RECNO | SMFSTRCN

SMF_STAT_RECNO specifies the SMF record number to use. The format is as follows:

```
SMF_STAT_RECNO ( record_number )
```

The *record_number* value is the number of the desired record of Oracle SMF statistics. The default is zero (0). Otherwise, the value must be specified between 128 and 255 for this parameter. Example:

```
SMF_STAT_RECNO(204)
```

The collection and writing of Oracle SMF statistics records is controlled by this single parameter in the OSDI service parameter file. A zero (0) for this parameter indicates that no SMF statistics record is to be written. The SMF record number that is chosen must not be the same as the number that is used by any other z/OS software.

If this parameter is not specified, or if zero is specified, then no SMF statistics collection or recording is done. This saves some CPU overhead and saves the overhead of the SMF write itself (which is mostly asynchronous work done by the SMF address space, the inline overhead is mainly just moving data into SMF buffers). For more information about SMF, refer to Appendix B, "The Oracle SMF Interface".

TRACE_DSNAME | TDSN

TRACE_DSNAME specifies the destination for gateway trace files. This includes normal traces requested by setting the TRACELVL environment variable as well as diagnostic traces generated automatically in certain error situations. The format is as follows:

```
TRACE_DSNAME ( filespec )
```

The *filespec* value is either a SYSOUT specification (including class, form, and JES destination) or a data set name.

A SYSOUT specification is of the form:

```
//SYSOUT:class,form,dest
```

When this is used, trace files are dynamically allocated SYSOUT data sets. In a multi-address space service, the trace file for a given database session is allocated in the address space that hosts the session. Thus, SYSOUT trace files can appear in all server address spaces. For example, traces written to SYSOUT class X, form AA01, would be written as:

```
TRACE_DSNAME(//SYSOUT:X,AA01)
```

For more information, refer to the *Oracle Database System Administration Guide for IBM z/OS (OS/390)*.

As an alternative to a SYSOUT specification, you can specify a data set name. Because each trace file created as a data set must have a unique data set name, the value supplied must include system symbols that guarantee uniqueness. For more information about system symbols, refer to the *Oracle Database System Administration Guide for IBM z/OS (OS/390)*

To guarantee uniqueness, use some combination of the session identifier (&ORASESST) system symbol, date (&LYYMMDD), and time (&LHHMMSS). Also, use high-level qualifiers that are appropriate for your installation. This will avoid the possibility of duplicating trace data set names generated in other Oracle instances you run. All components of the string must resolve to produce a name that is valid for a z/OS sequential data set. For example:

```
TRACE_DSNAME(DB2GW.TRACE.D&LYYMMDD..T&LHHMMSS..&ORASESST)
```

The allocation parameters for trace data sets are obtained from the DBTR file group of the server file management parameters. For more information about these parameters, refer to the *Oracle Database System Administration Guide for IBM z/OS (OS/390)*

If this parameter is omitted or fails to produce a valid, unique data set name, then all Oracle trace files are written to the default SYSOUT class associated with the server region.

G4DB2ENV (Environment Variables and HS Initialization parameters)

There are two types of parameters for member G4DB2ENV of the *db_hlq*.PARMLIB library:

- The first type is parameters that set an environment variable. An environment variable is used to direct internal gateway processing.
- The second type is heterogeneous services (HS) initialization parameters that are uploaded to the Oracle database server during the first connection of a session.

Parameters for Environment Variables in Member G4DB2ENV

Following are the valid parameters and their defaults for the environment variables in member G4DB2ENV of the *db_hlq*.PARMLIB library. Set these parameters to values that are applicable to your site:

Table 5–1 Valid Parameters for Environment Variables of Member G4DB2ENV

Parameters	Default
CURRDEGREE	1
DB2DESCTAB	YES
DB2LONGMSG	NO
DB2WARNING	NO
DB2STATS	NO
DB2READONLY	NO
FDS_CLASS_VERSION	10.2.0.2.0
FLUSH_CACHE_ON_COMMIT	NO
TARGET	DB2
DB2SSN	DSN0
DB2PLAN	G4DB2PLN
ORARECID	no default

Table 5–1 (Cont.) Valid Parameters for Environment Variables of Member G4DB2ENV

Parameters	Default
ORA_MAX_DATE	no default
ORACLE_TIMESTAMP	NO
TRACELEVEL	0
TO_NUMBER_OFF	NO
CNV_LIT_FMT	NO
EMPTYSTR_TO_NULL_OFF	NO
SELECT_CONCAT_ON	NO

CURRDEGREE

The CURRDEGREE parameter allows you to specify whether parallel query operations are to be allowed on your DB2 system. The valid values are:

Value	Description
1	turns off parallel query operations. The default value is 1.
ANY	specifies that parallel query operations are to be allowed on the DB2 system. Parallel processing must be enabled on the DB2 system before parallel query operations can occur.

For further information on DB2 parallel I/O and CP processing, refer to the IBM documents for your platform and operating system.

DB2DESCTAB

The DB2DESCTAB parameter specifies whether the gateway uses a DB2 DESCRIBE TABLE for acquiring DB2 table information. The default of DB2DESCTAB set to YES directs the gateway to run a DESCRIBE TABLE when acquiring information about the DB2 table. This can provide performance enhancements but cannot be used when accessing DB2 tables using a DB2 alias. DB2 does not allow a DESCRIBE TABLE against a DB2 alias. DB2DESCTAB set to NO prevents the DESCRIBE TABLE from being used by the gateway so applications can access DB2 tables using the DB2 alias.

DB2LONGMSG

The DB2LONGMSG parameter allows longer error messages to be returned by the gateway. The default of DB2LONGMSG set to NO returns some of the error messages from DB2 but does not always return the entire error message. By setting DB2LONGMSG to YES, you ensure that the entire DB2 error message can be returned by the gateway.

DB2WARNING

The DB2WARNING parameter enables you to configure the gateway so DB2 warning messages are not returned to the application as errors.

Warning messages are displayed by DB2 when SELECT statements are not formatted properly, but some default action by DB2 can still be taken to complete the task. Setting DB2WARNING to YES allows warning message to be returned by the gateway.

The warning message is interpreted as an error by the Oracle application. The default setting of DB2WARNING set to NO processes the SELECT statement without returning the warning to the application. The gateway continues DB2 default completion without notification to the application that a warning occurred.

DB2STATS

DB2STATS specifies whether or not the gateway is to pass DB2 statistics to the Oracle Optimizer for improved query performance. If set to the default value of NO, then the gateway does not pass statistics. If set to YES, then the statistics are used by the Oracle Optimizer to choose the access plan for SQL statements that involve DB2 objects. This results in DB2 tables appearing to the Oracle Optimizer as if they were Oracle-analyzed objects. If set to YES, then ensure that the DB2 utility RUNSTATS is run against the DB2 tables accessed by the gateway. This ensures information about these tables is available in the DB2 catalog. Also ensure that the gateway plan qualifier has SELECT privileges for the SYSIBM.SYSKEYS, SYSIBM.SYSINDEXES, and SYSIBM.SYSTABLES DB2 catalog tables.

DB2READONLY

DB2READONLY controls whether the gateway is enabled in read-only mode. If it is set to the default value of NO, then the gateway is not in read-only mode. If it is set to YES, then read-only capabilities are enabled and only queries are allowed through the gateway to DB2. Any SQL statements or calls that attempt to modify a

DB2 object are rejected. When the gateway is in read-only mode, INSERT, UPDATE, DELETE, DB2 stored procedures, or passthrough SQL are not allowed.

For additional information, refer to Chapter 8, "Using the Gateway".

FDS_CLASS_VERSION

This parameter controls AUTOREGISTER, the uploading of class capabilities from the gateway to the Heterogeneous Services layer of the Oracle Integrating Server. When the value of this parameter changes, it causes HS to upload and use the new G4DB2CAP capabilities table.

You should rely on the default and specify this parameter only at the request of Oracle Support Services.

FLUSH_CACHE_ON_COMMIT

FLUSH_CACHE_ON_COMMIT specifies when the describe table cache is flushed. If the value is set to YES, then the describe table cache is flushed each time a transaction is committed. If the value is NO, then the describe cache is flushed when the Oracle session terminates. The default setting of NO reduces the overhead associated with flushing cached information that is retrieved repeatedly after each committed transaction. Performance might be improved by using the default setting of NO.

The following parameters are set based on values specified during the installation and configuration process.

DB2CAP

With the introduction of the dynamic capability table, the DB2CAP environment parameter is obsolete.

TARGET

The TARGET parameter is maintained for backward compatibility with version 4 of the gateway. If you are not a previous customer of TG4DB2 v402110 using GTW_SQL.GTWPASS procedures, then you do not need to specify this parameter.

DB2SSN

This parameter specifies the DB2 subsystem name to be accessed by Oracle Transparent Gateway for DB2. The default parameter is DSN0.

DB2PLAN

The DB2PLAN parameter specifies the DB2 plan name that Oracle Transparent Gateway for DB2 uses to access the DB2 system. The default is G4DB2PLN.

ORARECID

This parameter specifies the user name that Oracle Transparent Gateway for DB2 uses when logging on to DB2 to perform recovery. This parameter has no default but must be defined for the gateway to initialize successfully.

ORA_MAX_DATE

This parameter must be specified as YYYY-MM-DD. The Oracle database server allows a maximum date value of 4712-12-31 while DB2 allows dates up to 9999-12-31. This ORA_MAX_DATE parameter has no default. If this parameter is specified, then any DB2 date value that is being sent back to the Oracle database server is inspected. If the inspected value exceeds 4712-12-31, then it is replaced by the value of ORA_MAX_DATE.

If no date value is specified for ORA_MAX_DATE, then a returned date value might not be valid to the Oracle database server. The value of ORA_MAX_DATE can be set to a value less than 4712-12-31, but the ORA_MAX_DATE is returned only if the DB2 date value exceeds 4712-12-31.

If you have no dates exceeding the year 4712, then you do not need to be concerned about this parameter.

Some examples are listed in the following table:

ORA_MAX_DATE	DB2 Value	Returned to Oracle
(not specified)	9999-12-31	9999-12-31 (invalid)
(not specified)	4713-10-24	4713-11-28 (invalid)
(not specified)	4500-11-19	4500-11-19
4712-12-31	9999-12-31	4712-12-31
4712-12-31	4713-10-24	4712-12-31
4712-12-31	4500-11-19	4500-11-19
3388-12-31	9999-12-31	3388-12-31
3388-12-31	4713-10-24	3388-12-31

ORA_MAX_DATE	DB2 Value	Returned to Oracle
3388-12-31	4500-11-19	3388-12-31

ORACLE_TIMESTAMP = YES| NO (default)

If set to YES, then DB2 timestamps are sent to the Oracle database server as Oracle TIMESTAMP(6) data types.

If set to NO, then DB2 timestamps are sent to the Oracle database server as CHAR(26). This is the default and the behavior prior to release 10*g*.

TRACELEVEL

If this parameter is set to 4, then the following information is written to the trace file:

- SQL text sent to the gateway by the Oracle database server
- SQL text sent to the target database

Set this parameter to 255 for all gateway trace information.

TO_NUMBER_OFF = YES | NO (default)

The fix for bug 2095461, "SQLCODE = -418 is returned using TO_NUMBER against a char bind variable", introduces a new initialization parameter. By default, a SQL statement sent through the gateway to DB2 that includes the TO_NUMBER function with one argument will be translated to the equivalent DB2 function, DECIMAL(). This can cause problems when the only argument is a bind variable. You can alleviate this problem by setting the new init parameter, TO_NUMBER_OFF to YES in the gateway ENV member.

This will turn off the capability, and the TO_NUMBER function will be processed by the Oracle instance prior to sending the SQL statement to the gateway.

CNV_LIT_FMT = YES | NO (default)

Oracle database and the gateway would perform implicit conversion when necessary before sending SQL to DB2. A specific case would be if there is a WHERE clause such as:

WHERE number_variable = 'literal_string'

where the literal string contains comma (,) as the decimal indicator. In that case, you would need to set CNV_LIT_FMT to YES.

A related bug of CNV_LIT_FMT is 1934416.

EMPTYSTR_TO_NULL_OFF = YES | NO (default)

When set to YES, HS will *not* convert empty string to NULL before sending to DB2.

A related bug of EMPTYSTR_TO_NULL_OFF is 2249392.

EMPTYSTR_TO_NULL_WHERE_OFF = YES | NO (default)

The gateway will translate an empty string in the WHERE clause to NULL by default. For example, the command `select * from emp@gtwy where ename = ''` would be translated to DB2 as `select * from emp where ename is NULL`. Oracle regards NULL and empty strings as the same, while DB2 distinguishes between them. If you specify EMPTYSTR_TO_NULL_WHERE_OFF = YES, then TG4DB2 will not attempt to translate the empty string.

A related bug of EMPTYSTR_TO_NULL_WHERE_OFF is 25887100

LIKE_OFF = YES | NO (default)

When set to YES, the gateway forces Oracle Integrating server to postprocess it.

A related bug of LIKE_OFF is 2528836

NVL_TO_VALUE_OFF = YES | NO (default)

The DB2 VALUE function cannot handle decimal bind variables. The gateway translates Oracle NVL to DB2 VALUE function by default. In a situation like bug 2359741, set NVL_TO_VALUE_OFF to YES.

A related bug of NVL_TO_VALUE_OFF is 2359741

SELECT_CONCAT_ON = YES | NO (default)

By default, the CONCAT function is post-processed. When this is set to YES, the CONCAT function is passed on to DB2 which may cause a SQLCODE -418 due to a DB2 restriction (bug 1269591).

HS Initialization Parameters for Member G4DB2ENV

Following are the valid HS initialization parameters that can be specified in member G4DB2ENV:

Table 5–2 Valid HS Initialization Parameters for Member G4DB2ENV

Parameters	Generic Default
HS_CALL_NAME	There is no default.
HS_DB_DOMAIN	WORLD
HS_DB_INTERNAL_NAME	DB21020
HS_DB_NAME	There is no default. Normally you would specify the gateway service SID.
HS_DESCRIBE_CACHE_HWM	100
HS_OPEN_CURSORS	50
HS_RPC_FETCH_REBLOCKING	ON
HS_RPC_FETCH_SIZE	4000 (Oracle recommends that this value be set to 40 000)

If GLOBAL_NAMES is set to true in the INIT.ORA file, then the HS_DB_DOMAIN parameter must match the DB_DOMAIN parameter in the INIT.ORA file.

Refer to the *Oracle Database Heterogeneous Connectivity Administrator's Guide* for additional information about the HS initialization parameters for member G4DB2ENV.

The following information varies from the information presented in the *Oracle Database System Administration Guide for IBM z/OS (OS/390)*. The information presented in this guide facilitates you to use the Oracle Transparent Gateway for DB2:

1. HS_DB_INTERNAL_NAME parameter default can be overridden. The default is DB21020.
2. There is no default for HS_DB_NAME. You could normally specify the gateway instance SID as HS_DB_NAME. When GLOBAL_NAMES is set to true in the Oracle integrating server, this parameter must match the name of the database link.
3. HS_LANGUAGE is obsolete and must not be specified. The appropriate setting is established automatically.

4. HS_NLS_DATE_FORMAT is obsolete and must not be specified.
5. HS_NLS_NCHAR is obsolete and must not be specified.
6. HS_RPC_FETCH_SIZE parameter defaults can be overridden. If the HS_RPC_FETCH_REBLOCKING parameter is set to ON (the default), then the array size for SELECT statements is determined by the HS_RPC_FETCH_SIZE parameter value. The recommended value for Oracle Transparent Gateway for DB2 is 40 000. The value shipped with Oracle Transparent Gateway for DB2 is 40 000.

   The HS_RPC_FETCH_SIZE parameter defines the number of bytes sent with each fetch between the gateway and the Oracle database server.

   Notes: This feature can provide significant performance enhancements, depending on your application design, installation type, and workload.

   The HS_RPC_FETCH_SIZE value impacts the performance for elapsed time.

   Generally, a higher HS_RPC_FETCH_SIZE value can correlate with improved performance for elapsed time.

   However, it is important to evaluate the requirements for elapsed time since a higher value associated with this parameter might also impact memory use.

Checklists

Checklists are provided for configuration and post configuration.

Configuration Checklist

Use the steps that follow to configure the gateway:

- Step 1: Run the Configuration Utility
- Step 2: Customize JCL Procedures and Parameter Files
- Step 3: Copy the Subsystem PROCs to a System PROCLIB
- Step 11: Define and Start OSDI Services
- Step 4: Make Authorization and Local Date Exits Available to DB2

Post Configuration Checklist

Creating a Gateway Instance

After installing the gateway software, you use the Oracle Universal Installer configuration utility to create one or more gateway instances. The configuration utility creates `INSTLIB` and `PARMLIB` libraries customized to your environment. Each gateway instance will have its own set of `INSTLIB` and `PARMLIB` libraries. These libraries contain the JCL procedures needed to configure the gateway instance.

Before using the configuration utility, determine the following information for the gateway instance to be created:

- Subsystem name of the gateway
- Names of the JCL procedures for the subsystem
- Gateway `SID`
- High-level qualifier for all PDS data sets
- Volumes on which PDS data sets will reside

This information should be determined in advance because it is used multiple times in creating the gateway files and can be more complicated to change later. For all other information, you can accept the defaults and change them manually, later.

The following steps provide guidelines for creating a gateway instance using the configuration utility. It is assumed that you have already installed the gateway software, performed the APF authorization, and put the necessary files in the linklist. If not, then you will need to complete those tasks before continuing. For more information, refer to Chapter 4, "Installation".

Step 1: Run the Configuration Utility

Start the Installer and select the Oracle Transparent Gateway for the DB2 configuration option. This starts the configuration utility which prompts for the following information:

- High-level qualifiers. Specify the high-level qualifiers for the location of the Oracle executable code and the gateway. Although you can use one high-level qualifier for both, it is recommended that you use a separate high-level qualifier for each.

  The high-level qualifier for the location of the Oracle executable code (*oracle_hlq*). For this high-level qualifier, you should include the version information, for example ORACLE.V10G.

  The high-level qualifier for the gateway (*db_hlq*) is used to identify the PDS data set files for the gateway instance. This high-level qualifier should be labeled with the gateway name or a similar name, for example, ORACLE.GTW1. It is recommended that you do not include version information in the gateway high-level qualifier, as this is likely to change over time.

- INSTLIB and PARMLIB libraries. Specify where to create the new INSTLIB and PARMLIB libraries. This can be done by using IBM Storage Management Subsystem (SMS) or by manually specifying a volume and unit.
- Subsystem Definition Parameters. Specify the subsystem name, gateway SID, net SID, port on which the gateway should listen for remote connection attempts, and names of the JCL procedures for the gateway.
- Gateway Parameters. Specify the basic OSDI parameter file definitions for the gateway.
- ORA$FPS Control File Definitions. Specify the information to create an ORA$FPS file. This file is used to create various files of a particular size. You can enter new values for your installation or enter default values and later modify the ORA$FPS file that is created.

The configuration utility generates two PDS data set files for the INSTLIB and PARMLIB libraries, *db_hlq*.INSTLIB and *db_hlq*.PARMLIB.

The INSTLIB library contains all the sample JCL required to configure the gateway instance and perform simple tasks like starting and stopping the gateway.

The PARMLIB library contains all the required parameter files needed to start the gateway instance.

Step 2: Customize JCL Procedures and Parameter Files

The degree to which the information provided in Step 1 is accurate will determine how much the JCL procedures and parameter files need to be modified in order to configure the gateway instance.

JCL Procedures

All JCL procedures (or batch jobs) need to be reviewed carefully to ensure that they are valid. The JOBCARD job provided is only a default used by Oracle and will need to be tailored to your environment.

The batch jobs can be divided into the following categories:

- Two sample PROCs that define the gateway address space and the Net address space, and a procedure to copy them into a system installation library.

  The PROCs are named in Step 1 of the configuration process and are cross-referenced to the OSDI parameter file. Each procedure must be defined to RACF as a started task and associated with a user.

  The batch job used to copy the two PROCs into a system PROCLIB library job is called COPYPROC and must be customized to point to the system PROCLIB library.

- The other batch job associated with this is called STRTSRVC. The STRTSRVC job is used to define and activate the subsystem. Another set of PROCs provide basic functions like starting the subsystem services.

Parameter Files

The parameter files are located in *db_hlq*.PARMLIB. They are the core definition files for the gateway. You should review these files for accuracy. The parameter files can be divided into the following categories:

- OSDI parameter files. Of this group of files, the subsystem definition file is the core file. This file is called only by the OSDI subsystem name. This file has a major impact on the other JCL batch jobs and parameter files and should be modified with care. The following is an example of the subsystem definition file:

```
INIT (ORASSI,SSN1)
DEF SVG SSID(SSN1) DESC('OSDI Gateway 10G Subsystem - SSN1')

DEF SRV DB2GW1 PROC(ODB2GW1) TYPE(GTW) -
 DESC('V10G GATEWAY Service') -
 SID(GTW1) PARM('ORACLE.ORA1.PARMLIB(G4DB2PRM)')

DEF SRV ORAN10 PROC(ORA1N10) TYPE(NET) -
 DESC('Oracle V10G Net Service') -
 SID(ORAN) PARM('HPNS PORT(1501) ENCLAVE(SESS)')

SHOW SERVICEGROUP LONG

START  DB2GW1
START  ORAN10
```

The file related to this is called `G4DB2PRM`. This contains all the OSDI specific parameters. These parameters are documented in detail in this guide. Most default parameters are acceptable for most basic installations. The following is an example of this file:

```
* SSN1 OSDI SUBSYSTEM PARAMETER FILE.
* USED BY SUBSYSTEM SSN1 SERVICE DB2GW1 PROC DB2GW1
* LOAD MODULE TO USE.
SERVER_LOADMOD(G4DB2SRV)
* NUMBER OF ADDRESS SPACES TO START. VALUES ARE 1-256
INIT_ADR_SPACES(1)
* MAXIMUM NUMBER OF SESSIONS ALLOWED FOR THIS ADDRESS SPACE
MAX_SESSIONS(500)
* MAX MEMORY ALLOWED PER SESS. VALUES ARE NNNN {K|M}.
MAX_SESSION_MEM(100M)
* INITIAL STACK SIZE. VALUES ARE NNNN {K|M}.
INIT_STACK_SIZE(128K)
* TURN ON SMF RECORDING. VALUES 0 AND 128 THROUGH 255.
SMF_STAT_RECNO(0)
* EXTERNAL AUTHENTICATION
LOGON_AUTH(G4RRSAF)
* STORAGE CUSHION. VALUES ARE NNNN {K|M}.
REGION_MEM_RESERVE(10M)
* DATABASE DATASET NAME PREFIX.
DSN_PREFIX_DB(G4DB2)
* TRACE DATASET NAME PREFIX.
TRACE_DSNAME(ORACLE.GTW1.TRACE.&ORASESST..T&HHMMSS)
* ALERT DATASET NAME PREFIX.
ALERT_DSNAME(ORACLE.GTW1.ALERT.&ORASESST..T&HHMMSS)
```

The third file in this section is called SUBSYS. It provides the command which needs to be issued in order to define and activate the subsystem definition file.

- The G4DB2FPS parameter file contains the default FPS parameters for creating the trace files. Remove any which are not required. If this is defined correctly, then defining additional tran files becomes easier. The minimum that is recommended is DFLT. Refer to this guide for details about what to code for this parameter file. An example is as follows

```
* Default parameters
FILE_GROUP(DFLT)
  RECALL(NONE)
  MOUNT(NO)
  DEFAULT_SPACE(10000 10000)
  UNIT(SYSDA)
```

- Instance-specific files

  The G4DB2ENV contains the environment variables required for the gateway.

  Two other files which can be used are the TNSNAMES file and the SQLNET file. The TNSNAMES file provides the default TNSNAMES entries for users to access this instance both through cross memory and through TCP/IP. This can be used by the Oracle database server than connects to this gateway.

Step 3: Copy the Subsystem PROCs to a System PROCLIB

When the `COPYPROC JCL` has been customized correctly and the `PROC` names defined for the database are valid, run the `COPYPROC JCL`. It copies the two subsystem `PROC`s into a system `PROCLIB` library.

Step 4: Make Authorization and Local Date Exits Available to DB2

If a local date exit is required, then copy the `DSNXVDTX` exit to the DB2 exit library. The gateway exits are in the `AUTHLOAD` library.

The DB2 Local Date Exit (`DSNXVDTX`) provided with the gateway supports Oracle date formats `DD-MON-RR` and `DD-MON-YYYY`. This requires a DB2 installation update to set the DB2 local date length. Option 10 on DB2 install panel `DSNTIPF`, which specifies the local date length, must be set to 11, if not already set as 11.

You must stop and start the DB2 subsystem to access the DB2 local date exit and activate the DB2 local date length parameter.

Step 5: Run the Scripts to Create Required Tables and Views in DB2

There are now 4 versions of the DB2 views. If you use DB2 7.1, then use one of the DB2 7.1 SQL scripts. If you use DB2 8.1, then use one of the DB2 8.1 SQL scripts.

The "external security" views are if you use DB2 external security. This DB2 feature calls an external exit which typically makes RACF calls to see if a user has access to a DB2 object (rather than checking against GRANT information in the DB2 catalog).

If you use GRANTs in DB2 to provide access to DB2 objects (such as tables), then use the "standard security" views for your release.

Note that the external security views show all objects of a particular type. For example ALL_TABLES shows all tables in the DB2 catalog, where the standard security views only show the tables you have been granted access to through the DB2 GRANT command. Since DB2's external security uses an exit to check a user's access to DB2 objects (like tables), the GRANT information in the DB2 catalog is ignored, and so can't be used to verify if a user has access to an object.

The external security views allow a user to see the definition of all of the objects in the DB2 catalog. However, access to the objects is still restricted by DB2, and a user can still only access objects that they have access to.

The SQL scripts available are:

```
G4DDVWS7       DB2 7.1 Standard Security views
G4DDVWR7       DB2 7.1 External Security views
G4DDVWS8       DB2 8.1 Standard Security views
G4DDVWR8       DB2 8.1 External Security views
```

For the DB2 database named in the configuration, use the DB2 SPUFI utility or batch job to run the following SQL scripts for all gateway installations:

- *db_hlq*.INSTLIB(G4DB2DDT)
- *oracle_hlq*.SRCLIB(G4DDVWxx) where xx is one of the SQL Scripts listed above (S7, R7, S8, R8)

> **Attention:** If you migrated from a lower release of Oracle Transparent Gateway for DB2, then you must first run *db_hlq*.INSTLIB(G4DB2DLV). Refer to Chapter 11, "Migration and Coexistence with Existing Gateways", for more information.

Step 6: Bind the DB2 Package

Oracle Transparent Gateway for DB2 requires you to bind gateway DBRM G4DB2PLN to 10 separate packages. The 10-package mechanism enables a maximum of 5000 DB2 cursors per session while keeping EDMPOOL usage under control.

When you bind a package, you specify the collection to which the package belongs. These 10 packages should be included in the package list when binding the gateway plan. You can bind the packages in either of two ways:

- In a batch job, with the JCL supplied in the G4DB2BPL member of the *db_hlq*.INSTLIB library. The JCL supplied in this member performs both the bind package and the bind plan.
- Interactively, using DB2I.

  Use the IBM DB2I Bind Package panel options to bind each package. This step must be performed 10 times, once for each collection ID (G4DB2V102021A through G4DB2V102021J). The panels that are discussed below are for DB2, release 6.1, and release 7.1. In some cases, improved performance can be achieved by selecting different options from those discussed. Consult with your DB2 administrator to determine which options best suit your particular environment.

On the DB2I Bind Package panel, use the suitable settings for your installation. Following is a description of each option:

- Option 1 specifies which DB2 system to use to bind the package. If left blank, then this option defaults to the local DB2 system.
- Option 2 specifies the DB2 collection in which the package is located. The following naming convention must be used for the collection ID: G4DB2V102021x, where x is any letter in the range A through J. The bind package must be performed 10 times, once for each collection ID from G4DB2V102021A through G4DB2V102021J.
- Option 3 specifies whether you are creating a new package or making a copy of an existing package. You must specify DBRM for this option.
- Option 4 specifies which DBRM member to bind. You must specify G4DB2PLN for this option.
- Option 5 specifies a password for the library name listed in the LIBRARY field. You can leave this option blank.

- Option 6 specifies the name of the library containing the DBRM specified in the MEMBER field, G4DB2PLN. In this example, the DBRM is located in data set *oracle_hlq*.SRCLIB.
- Option 7 specifies whether to change current defaults. You must specify YES to be able to change current defaults.
- Option 8 specifies whether to enable or disable other IBM intersystem connection types to use with this package. You should specify NO for this option.
- Option 9 specifies the primary authorization ID owning the package.
- Option 10 should be left blank so that the qualifier specified on the Bind Plan panel takes precedence.
- Option 11 specifies whether to replace an existing package or add a new one. You should specify REPLACE for this option.
- Option 12 specifies whether to replace a specific version of the package or create a new one. You should leave this option blank.

For additional information about the options for the Bind Package panel, refer to the IBM documents for your platform and operating system.

When you complete the changes on the Bind Package panel, press **Enter** to continue.

- Option 1 specifies the isolation level of the package. You should leave this option blank so that the isolation level specified on the Bind Plan panel takes precedence.
- Option 2 specifies when to validate DB2 objects or privileges for the package. You should specify BIND for this option.
- Option 3 specifies when to release locks on resources. You should leave this blank so that the resource release time specified on the Bind Plan panel takes precedence.
- Option 4 specifies whether to obtain EXPLAIN information on how SQL statements in the package run. You can leave this option blank.
- Option 5 specifies whether data currency is required for ambiguous cursors when the isolation level of cursor stability (CS) is in effect. This option also determines whether block fetching can be used for distributed, ambiguous cursors.

Although cursors used in block fetch operations result in reduced network traffic, you are still vulnerable to reading data that has already changed. In a block fetch, DB2 fetches as many rows as can fit into a buffer before sending the entire buffer over the network. During that time, the underlying data might have been modified before the application actually asks for the data. Situations such as fetching a row of values that no longer exists, or missing a recently inserted row, can occur. If these types of situations are acceptable, then data currency is not required (`CURRENTDATA=NO`). If data currency is required (`CURRENTDATA=YES`), then be aware that a separate buffer is sent over the network for each row fetched.

- Option 6 specifies the use of parallel processing (if possible) to run queries. You can leave this option blank.
- Option 7 specifies whether run-time rules or bind-time rules apply to dynamic SQL statements at run time. You should specify `RUN` for this option.
- Option 8 should be left blank.
- Option 9 specifies whether to have DB2 determine an access path at run-time using values for host variables, parameter markers, and special registers. You should set this option to `NO`.
- Option 10 specifies whether to defer preparation for dynamic SQL statements that refer to remote objects. You should set this option to `NO`.
- Option 11 determines whether DB2 keeps dynamic SQL statements after commit points. You should set this option to `NO`.

For additional information about the options for the Defaults for Bind Package panel, refer to the IBM documents for your platform and operating system.

When you complete the changes on the Defaults for Bind Package panel, press **Enter** to continue. Repeat this step, varying only the collection ID, until all packages (`G4DB2V102021A` through `G4DB2V102021J`) are bound.

Step 7: Bind the DB2 Plan

You can bind the gateway plan in either of two ways:

- As a batch job

  Use a batch job with the `JCL` supplied in the `G4DB2BPL` member of the Oracle *`db_hlq`*`.INSTLIB` library. The JCL supplied in this member performs both the bind package and the bind plan.

Note: If the job has run successfully in "Step 6: Bind the DB2 Package", then it does not need to be run again.

You can proceed to "Step 8: Grant `EXECUTE` on DB2 Plan".

- Interactively, using DB2I

  Use the IBM DB2I Bind Plan panel options to bind the plan. (The panels discussed are for DB2 release 7.1.) In some cases, improved performance can be achieved by selecting options different than those discussed in this section. Consult with your DB2 administrator to determine which options best suit your particular environment.

  On the IBM DB2I Bind Plan panel, use the appropriate settings for your installation. The options are:

- Option 1 specifies the first DBRM to include in the gateway plan. Specify `G4DB2IX`, `G4DB2PRC`, and `G4DB2V51` for this option.
- Option 2 specifies the password for the libraries listed in the `LIBRARY` field. You can leave this option blank.
- Option 3 specifies the name of the PDS containing the gateway DBRMs. In this example, the DBRMs are located in data set *`oracle_hlq`*`.SRCLIB`.
- Option 4 must be set to `YES` to specify additional DBRM members.
- Option 5 specifies the name of the DB2 application plan to create. Specify the DB2 plan name entered on the gateway configuration panel. The default plan name is `G4DB2PLN`.
- Option 6 must be set to `YES` to change current defaults.
- Option 7 specifies whether to enable or disable other IBM intersystem connection types to use with this package. Specify `NO` for this option.
- Option 8 specifies whether to include a package list for the plan. You must specify `YES` for this option.
- Option 9 designates the primary authorization ID owning the plan.
- Option 10 specifies the user ID from the gateway installation panel that is used as the qualifier. This value must be set to `OTGDB2`.

- Option 11 specifies the size (in bytes) of the authorization cache. For details about this option, refer to the IBM documents for your platform and operating system.
- Option 12 specifies whether this plan is new or is being replaced. Specify REPLACE for this option.
- Option 13 determines whether users with the authority to bind or run the existing plan are to keep that authority over the changed plan. Specify YES for this option.
- Option 14 specifies the initial server to receive and process SQL statements. You can leave this option blank.

For additional information about the options for the Bind Plan panel, refer to the IBM documents for your platform and operating system.

When you complete the changes on the Bind Plan panel, press Enter to continue.

Confirm the DBRM library. Press the `PF3` key to continue.

The IBM DB2I Defaults panel lets you set the `ISOLATION LEVEL` and `RESOURCE RELEASE TIME` to those values applicable to your site. `ISOLATION LEVEL` specifies the ways in which read-only operations are isolated from the effects of concurrent write operations.

The gateway does not change the locking behavior of the DB2 database. Therefore, it is important to understand how the `ISOLATION LEVEL` can impact the behavior of Oracle applications accessing DB2 through the gateway.

Oracle applications written for the Oracle database server and deployed later to access DB2 through the gateway might require special consideration. Oracle read operations do not prevent concurrent write operations and, therefore, do not have to `COMMIT` in order to allow subsequent updates to the Oracle database server. Thus, many Oracle applications do not `COMMIT` at the end of read operations.

DB2 generally requires a `COMMIT` to release read transaction locks to allow subsequent write operations by other DB2 transactions. If a previously developed Oracle application that does not run a `COMMIT` statement after reading data is redirected to a DB2 database, then DB2 might prevent subsequent DB2 transactions from updating the data. This is because, unlike the Oracle database server, DB2 acquires locks when data is read.

Refer to the IBM documents for your platform and operating system before setting the `ISOLATION LEVEL` to one of the following settings:

- Cursor stability (CS) can be chosen to work with the gateway for maximum concurrency with data integrity. However, if a read-only application does not COMMIT or close the cursor, then the CS isolation level might result in read locks being set that would prevent concurrent and subsequent write operations.
- Uncommitted read (UR). This isolation level can be used with the gateway to allow Oracle applications that do not COMMIT with read transactions to access DB2 without preventing write operations. It is important to note that the UR isolation level allows gateway applications to read uncommitted data.
- Repeatable read (RR) results in locks being held on all rows or pages accessed, preventing concurrent and subsequent write operations until the application performs a COMMIT.

Ensure that the other parameters are specified as listed.

On the IBM DB2I Package List for Bind Plan panel, fill in the collection names and package ids to include in the gateway plan. If you plan to run DB2 stored procedures through the gateway, then insert an asterisk (*) for the collection ID and package ID as the last entry in the COLLECTION and PACKAGE-ID columns.

When you complete the changes on the Package List for Bind Plan panel, press the PF3 key to bind the plan.

Step 8: Grant EXECUTE on DB2 Plan

Using the DB2 SPUFI utility, grant EXECUTE authority to public on the DB2 plan by running the G4DB2GNT member in the gateway *db_hlq*.INSTLIB library.

Step 9: Edit the PARMLIB Members

Edit the members of the PARMLIB library to ensure the gateway parameters are set appropriately for the installation. For the list of PARMLIB members, refer to"G4DB2ENV (Environment Variables and HS Initialization parameters)" on page 5-15.

The ORA$FPS parameter file contains file group definitions which are specified using keyword(value) syntax. Each definition must start with the keyword FILE_GROUP(name) and continue until the next FILE_GROUP keyword is encountered. Comments must start with an asterisk (*) and can begin in any column as long as comments (that are on the same line as keywords) are separated from the last keyword by at least one blank.

Keywords can be coded one per line or strung together on the same line separated by at least one blank, but a keyword (value) pair cannot be split across two lines.

No defaults are defined for the parameters. The default file group (DFLT) supplies parameters for any file group that is completely omitted from the file management parameters.

With this release, the only File Groups supported by or applicable to the gateway are DBTR, indicating the attribute for gateway trace files, and NTTR (network trace).

DATACLAS (classname)

Specifies an SMS data class name to be specified on `DEFINE CLUSTER` or dynamic allocation requests to create new data sets. `DATACLAS` can be abbreviated DATACL.

DEFAULT_SPACE (primary secondary)

Specifies default primary and secondary space quantities for a data set that is being created. The secondary quantity is optional and is ignored at this time. Both values must be numbers and are expressed in kilobyte (1024-byte) units. `DEFAULT_SPACE` can be abbreviated SPA.

FILE_GROUP (name)

Specifies the file group to which the file management parameters belong, where name is one of the allowed 4-letter file group names. This ends any in-progress file group definition and begins a new one. `FILE_GROUP` can be abbreviated FILE.

MGMTCLAS (classname)

Specifies an SMS management class name to be specified on DEFINE CLUSTER or dynamic allocation requests to create new data sets. `MGMTCLAS` can be abbreviated MGMTCL.

STORCLAS (classname)

Specifies an SMS storage class name to be specified on `DEFINE CLUSTER` or dynamic allocation requests to create new data sets. `STORCLAS` can be abbreviated STORCL.

UNIT (unitname)

Specifies an allocation unit name to use in dynamic allocation requests that create new non-VSAM data sets.

VOLUMES (volser)

Specifies a volume serial number to use in `IDCAMS DEFINE CLUSTER` commands for VSAM data sets or in dynamic allocation requests that create non-VSAM data sets.

Only a single volume serial can be specified. Because of this limitation, it is recommended that you use storage management class parameters instead of explicit volumes. `VOLUMES` can be abbreviated VOL.

Example:

Storage management parameter example:

```
* Oracle gateway file management parameters
* Trace data files
FILE_GROUP(DBTR)
DEFAULT_SPACE(200 50)                * a comment
* Default for groups not specified
FILE_GROUP(DFLT)
DEFAULT_SPACE(10000   5000)
UNIT(SYSDA) VOL(TEMP01)
```

SQLNETLG

This DD statement is optional and is the default destination for network error messages. `SYSOUT` or a sequential disk data set can be specified.

Step 10: Associate User IDs with Services

Services that are managed by Oracle Database 10*g* execute as system address spaces, similar to started tasks or STCs. Some of the OS/390 system functions that are called by Oracle Database 10*g* services perform authorization checks based on the OS/390 user ID that is associated with the service address space. Depending on the security configuration and standards of your installation, those system functions may fail if no user ID is associated with the address space. You, or security personnel for your installation, may need to take steps to ensure that Oracle Database 10*g* services have an associated user ID that can be authorized for system functions that are called by the database and network services.

With RACF, this authorization is normally accomplished by defining appropriate profiles in the STARTED resource class. Each profile associates a RACF user ID with a started task based on the JCL procedure name of the started task. You will choose and specify JCL procedure names for Oracle Database 10*g* services when those services are configured. You may want to decide on procedure names now,

however, so that RACF profiles can be defined. There are no special procedure naming requirements as far as Oracle Database 10*g* is concerned, so you can choose procedure names that meet the standards or requirements of your installation. Of course, the names should not be the same as the names of any members already in your system procedure library. The ISPF panel-driven gateway configuration process will generate a JCL procedure name based on the service SID specified. The generated name is G4*sid*, where *sid* is the service's SID. After the configuration process is complete, you may change the name of this procedure if it does not match the standard naming conventions at your site.

Defining the JCL procedure name in the `USER` resource class is an alternate method in which the procedure name itself is also used as the userid.

If you are already running the OSDI and TNS programs as started tasks (as opposed to submitting them as batch jobs), then your installation probably already has `STARTED` or `USER` profiles for the associated JCL procedures. You should not rely on those for Oracle Database 10*g* because the Oracle Database 10*g* procedures should have different names. Plan to create at least two new `STARTED` or `USER` profiles, one for the gateway service and one for the network service. These may be all that you need, because different instances of a type of service can generally share the same JCL procedure. You may want to create additional profiles, though, if you want different instances of a service to run with different user IDs. Note that this requires using distinct JCL procedures even though the procedures themselves may be otherwise identical.

Details on the `STARTED` and `USER` resource classes are in the RACF System Administration Guide. The `RDEFINE` command that is used to add profiles is described in the *RACF Command Language Reference*.

With RACF, it is possible to associate a user ID with a started task using a started procedures table that is built with Assembler macros somewhat like the resource class table discussed in the previous section. Activating such changes requires an IPL, however, and is not the preferred method. Refer to the *RACF System Administrator's Guide* for more information.

Step 11: Define and Start OSDI Services

This can be done by running the STRTSRVC member. This uses the SETSSI command to define the OSDI subsystem. If the user is not able to issue this command then the define service needs to be done by a system programmer. A sample of the command to issue is provided by the SUBSYS file in the PARMLIB.

At the end of this step, you should have a gateway and a Net subsystem defined and active.

Step 12: Start the Gateway

You can start the gateway service with OSDI Start Command. For example, from SDSF, run:

```
/G4XX START GTW1
```

where G4*XX* is the OSDI subsystem name and GTW1 is the service name.

Post-configuration Steps

The following optional steps can be performed any time after the gateway is configured.

Step 1: Move Reentrant Modules to z/OS Link Pack Areas

Step 2: Examine Oracle Dump Data Sets and Modify as Necessary

Step 3: Examine Oracle Trace Data Sets and Modify as Necessary

Step 1: Move Reentrant Modules to z/OS Link Pack Areas

The Oracle AUTHLOAD modules that have RMODE set to ANY and are reentrant can be placed in the z/OS extended pageable link pack area (EPLPA) to decrease storage requirements. Other modules that are linked with RMODE set to 24 and are reentrant can be placed in the z/OS pageable link pack area (PLPA) below the 16M line. For modules used by multiple batch or TSO users concurrently, real storage working set requirements are greatly reduced because all users of a given module share the same copy.

Following are some considerations for placing Oracle modules in z/OS link pack areas:

- A z/OS IPL is generally required to add, remove, or replace a module in the z/OS link pack areas. This complicates the timely application of maintenance or fixes.
- It might be necessary to move a module from the Oracle AUTHLOAD library to another z/OS data set so it is accessible for z/OS link pack area placement.
- Adding modules to the PLPA reduces the maximum private area size of all z/OS address spaces. This impact must be evaluated before moving new modules.

For details about adding modules to z/OS link pack areas, refer to the IBM documents for your platform and operating system. Details regarding which

modules are candidates for z/OS link pack area placement are covered in the *Oracle Database System Administration Guide for IBM z/OS (OS/390)*.

Step 2: Examine Oracle Dump Data Sets and Modify as Necessary

When the gateway encounters an abend in one of its tasks, it dumps the abend to a `SYS1.DUMP` data set. Because the gateway does not attempt to dynamically allocate dump data sets, you must ensure that a `SYS1.DUMP` data set is always available. The `SYS1.DUMP` data set must be large enough to hold two address spaces (the Oracle address space if there is one and the gateway address space). Refer to the IBM documents for your platform and operating system for information about managing dump data sets.

If a `SYS1.DUMP` data set is not available when needed, then z/OS directs OSDI dumps to a `SYSMDUMP DD` statement if coded in the OSDI Gateway startup JCL.

If a `SYSMDUMP DD` statement is directed to `SYSOUT`, then multiple dumps are preserved (at the cost of potentially large amounts of SPOOL space). The z/OS external writer must be used to extract such dumps from the SPOOL file. This file can be written to a tape or DASD data set using an IBM external writer. For information about using the external writer program, refer to the IBM documents for your platform and operating system.

Do not specify `SYSUDUMP` and `SYSABEND` in OSDI Gateway startup JCL, because they produce dumps that are not computer-readable.

Step 3: Examine Oracle Trace Data Sets and Modify as Necessary

The gateway attempts to gather as much information as possible when internal errors occur for a user or process and when gateway tracing is turned on through the `TRACELEVEL` parameter in the gateway environment `PARMLIB` member `G4DB2ENV`. This information is placed in an Oracle trace data set. The trace data set name is generated based on the OSDI `TRACE_DSNAME` value defined in the OSDI Gateway Region parameters in `PARMLIB` member <service_name>PARM.

The `TRACE_DSNAME` value can specify either a `SYSOUT` specification or a data set name. For complete information and syntax of this parameter, see "TRACE_ DSNAME | TDSN" on page 5-14.

6

Oracle Net

Oracle Net for z/OS supports network communications between Oracle applications and Oracle gateway systems across different z/OS systems and different operating systems. Oracle provides OSDI listener (ORANET). This chapter describes how to configure it. For more information on Oracle Net, refer to the *Oracle Database Net Services Administrator's Guide*.

This chapter includes the following sections:

Overview

The OSDI listener (ORANET), also referred to as the Net service, runs as a service under an OSDI subsystem. In Oracle8*i*, Release 8.1.7 and Oracle9i, Release 1, all TCP and LU62 connections by Oracle applications, both client and server, were performed through the Net or Net8 service. Now, all Oracle clients on z/OS open their own sockets.

The OSDI listener's primary function is to listen for inbound remote connections to an Oracle instance. For compatibility purposes, the OSDI listener still provides outbound connectivity services for Oracle9*i*, R1 and Oracle8*i*, 8.1.7 Oracle clients.

Note that in the case of a database link from an OSDI database instance to an OSDI gateway instance, the OSDI database instance is considered as an Oracle client in the context of OSDI listener for the Oracle gateway instance. Moreover, if the OSDI database instance resides in the same OS390 image as the OSDI gateway instance, then Oracle Net cross-memory protocol could be used.

An OSDI listener is not required for Oracle Net cross-memory protocol. The syntax of cross-memory protocol could be in either of two forms:

```
(DESCRIPTION=
    (ADDRESS= (PROTOCOL=XM) (SUBSYS=snn) (SERVICE=service_name) )
    (HS=)  )
```

where *ssn* is the OSDI subsystem name and *service_name* is the OSDI gateway service name.

or:

```
(DESCRIPTION=
    (ADDRESS= (PROTOCOL=XM) (SID=gateway_sid) )
    (HS=)   )
```

where *gateway_sid* is the chosen OSDI gateway service SID.

Oracle recommends using the SID format for simplicity.

OSDI Listener Architecture

On z/OS, Oracle Net is implemented as a z/OS OSDI service running in its own address space separate from the gateway service. The OSDI service acts as a listener for the Oracle z/OS instances and gateway instances. All protocol-specific code runs inside the OSDI listener.

Remote clients that access an OSDI service through an OSDI listener are dispatched on a lightweight unit of work called an enclave SRB. An enclave is created either once per session or for each SRB depending on the `ENCLAVE` keyword (described under "PARM" on page 6-5). An SRB is scheduled each time work is required to be done by the kernel. The enclave is deleted when the SRB completes. The z/OS Workload Manager component may be used to control the execution characteristics of these enclave SRBs. Refer to the *Oracle Database System Administration Guide for IBM z/OS (OS/390)* for further details.

For client and server support, the OSDI listener uses the IBM macro implementation and a TCP/IP network to support network communications between the Oracle

server and any remote OSDI listener TCP/IP client or server. For more information, refer to "TCP/IP Network Considerations" on page 6-8.

OSDI Listener File Names

The product documentation, *Oracle Database Net Services Administrator's Guide*, refers to files in the following form:

basename.extension
where:

Variable	Description
basename	is the product name.
extension	is the extension.

An example of this form is SQLNET.ORA.

These files are then converted to DD names. The following DD names are implemented under z/OS:

DD name	Description
SQLNET	defines a data set containing any SQLNET.ORA diagnostic, ASO, or Oracle names parameters. It is not necessary to allocate this DD unless these features are desired. Refer to *Oracle Database Net Services Administrator's Guide* or the *Oracle Database Advanced Security Administrator's Guide* for more information.
SQLNETTC	defines a data set into which trace output is written. It is recommended that this be defined as a SYSOUT data set in a held output class.
SQLNETLG	defines a data set into which any logging output is written. It is recommended that this be defined as a SYSOUT data set in a held output class.
TNSNAMES	defines a data set containing all the TNS connect descriptors and aliases for your installation. For further information about TNS connect descriptors, refer to the *Oracle Database Net Services Administrator's Guide*. This DDname is not necessary on server JCL unless DBLINKS originates from the server.
LDAP	defines the location of the LDAP server.
TNSNAV	TNS client navigation. (Generally not used on z/OS.)

DD name	Description
INTCHG	Interchange. (Generally not used on z/OS.)

Example of diagnostic entries in SQLNET file

```
trace_file_client =/trace/sysout=x,hold
trace_file_server  =/trace/sysout=x,hold
trace_file_agent   =/trace/sysout=x,hold
trace_level_client  = 16
trace_level_server  = 16
trace_level_agent   = 16
trace_functions_all = yes
```

Example of TNSNAMES entry for use by DBLINK definition

```
G4XXDB2 =
 (DESCRIPTION =
    (ADDRESS=(PROTOCOL=TCP)(HOST=your.host.tcpip.name)
              (PORT=port#)(SSN=net_sid))
              (HS=)
              (CONNECT_DATA=(SID=G4XX))
```

Configuring the OSDI Listener

To create a Network Service under OSDI, you must first define the OSDI listener as a service using the `OSDI DEFINE SERVICE` command. In addition to defining the service, two other items must be set up before the service can be started. They are a JCL procedure and network protocol-specific (TCP/IP) configuration. After you have defined OSDI listener as a service and have set up the additional items, you can start the service, which creates a z/OS address space based on controls that you have specified.

Network Service Definition

The OSDI `DEFINE SERVICE` command is described completely in Appendix B. Below, we describe `DEFINE` parameter considerations that are specific to the OSDI listener.

Service Name

The service name for Oracle Net can be anything that you want within the content limitations described in Appendix B.

TYPE

The TYPE parameter for a gateway service must be specified as NET.

PROC

This procedure specifies the name of a service JCL procedure that you will place in one of your system procedure libraries. The procedure need not exist when DEFINE SERVICE is issued, but it must be in place before the service is started. The procedure name can be anything that you choose or that the naming standards of your installation require. The requirements for this procedure are discussed in the "OSDI Listener Region JCL"section on page 6-6.

PARM

The PARM string is used to specify additional initialization parameters that are specific to the OSDI listener. These parameters are in the form of keywords and determine which protocols are initialized at OSDI listener startup as well as configuration and debugging features.

A description of the OSDI Listener keywords follows:

OSDI Listener Keywords	Description
HPNS	specifies support for the TCP/IP protocol.
ENCLAVE(SESS|CALL)	specifies the duration of the enclave. When SESS is specified, the enclave is created at logon and deleted at logoff. When CALL is specified, the enclave is created when the server is sent a request and is deleted when the server waits for a receive.
PORT(nnnn)	specifies the TCP/IP port number (*nnnn*) on which to listen for incoming connections. The default is 1521.
GTF	may be specified at the request of Oracle Support Services. This allows the OSDI listener internal trace to be captured to the z/OS Generalized Trace Facility.
DUMP(nodename)	specifies the high level node, or nodes, of transaction dump data set names. The character string can be up to 26 characters in length, must follow the rules for z/OS data set names, and must not end with a period. When an OSDI listener transaction dump occurs, the value defined here will be prefixed to a string that includes a time and date stamp to generate a unique data set name. The default is ORACLE.TRANDMP.

Example of Network Service Definition

```
DEFINE SERVICE NET92 TYPE(NET) PROC(ORANET92) -
DESC(' Oracle Network Service 9.2') -
SID(NETG) -
PARM('HPNS GTF PORT(1521)) DUMP(ORACLE.TRANDMP)')
```

Note: The entire `PARM()` string must be on one line.

Note: if you have an active OSDI database instance, you could use an existing OSDI subsystem for the Oracle gateway instance and/or use existing NET service instance for the Oracle gateway instance.

OSDI Listener Region JCL

As with a gateway service, a JCL procedure must be placed in a system procedure library prior to attempting a start of the service. The OSDI listener JCL procedure name must have an associated z/OS user ID. Refer to the next topic, "TCP/IP Network Considerations" on page 6-8 for details. The `EXEC` card of the JCL must be equivalent to the following:

```
//NET     EXEC PGM=ORANET,REGION=0M
```

`REGION=0M` is specified to ensure that the service can allocate as much private virtual memory as it needs. Some z/OS systems may prohibit or alter a `REGION` parameter such as this, so you might want to check with your systems programmer to determine if any changes must be made to allow the system to accept your `REGION` parameter. In addition, the following DD statements are required:

STEPLIB: This DD statement should point to the Oracle-supplied `AUTHLOAD` which contains the gateway and Net load modules.

NET8LOG: Connection-related informational messages, warning messages, and error messages are written to this sequential output file. Oracle recommends that it also be assigned to a JES spool file.

Note: The OSDI listener JCL procedure name must have an associated z/OS user ID. Refer to the next topic, "TCP/IP Network Considerations", for details.

Example of Network Service Procedure JCL

```
//NET EXEC PGM=ORANET,REGION=0M
//STEPLIB DD DSN=ORAN.ORAV.AUTHLOAD,DISP=SHR
//NET8LOG DD SYSOUT=X
```

Example of NET8LOG output

```
2000034 09:50:35.0 MIN0017I message service subtask initialized
2000034 09:50:35.0 MIN0016I command service subtask initialized
2000034 09:50:35.1 MIN0018I bind/unbind service subtask initialized
2000034 09:50:35.2 MIN0026I timer service subtask initialized
2000034 09:50:35.2 MIN0002I networking service NETC     initialization complete
2000034 09:50:35.2 MIN0005I global vector is at 19F0A000
2000034 09:50:35.2 MIN0024I connected to WLM subsystem OSDI
2000034 09:50:50.4 MIN0700I HPNS INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:50.5 MIN0724I HPNS GHBY INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.1 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.1 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.1 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.2 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.2 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.2 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.2 MIN0713I I am listening on port 01522 socket 00000
2000034 10:05:58.8 MIN0733I Socket 0000 connected Subtask Kid1, IP 144.025.040.217, Port 01129.
2000034 10:05:58.8 MIN0733I Socket 0000 connected Subtask Kid2, IP 144.025.040.217, Port 01130.
2000034 12:00:13.9 MIN0098I networking service NETC     termination in progress
2000034 12:00:18.9 MIN0722I HPNS Kid #003 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #001 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #006 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #002 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #005 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #004 shut down.
2000034 12:00:18.9 MIN0723I HPNS Gethostbyname subtask ended.
2000034 12:00:18.9 MIN0721I HPNS shut down, GoodBye.
2000034 12:00:18.9 MIN0091I timer service subtask terminated
2000034 12:00:18.9 MIN0095I bind/unbind service subtask terminated
2000034 12:00:18.9 MIN0093I command service subtask terminated
2000034 12:00:18.9 MIN0094I message service subtask terminated
MIN0000I End of Net8 Log.
```

TCP/IP Network Considerations

The OSDI Listener uses the MACRO API interface for TCP/IP, and distributes the communications processing workload across multiple tasks in the OSDI listener address space.

If the IBM stack is being used, then particular attention must be paid to the `MAXFILEPROC` and `MAXSOCKETS` parameters (under AF_INET) in the `BPXPRM`*xx* member of `SYS1.PARMLIB`. These parameters must be set high enough to support the expected connection load. Both of these parameters can limit the number of connections that the OSDI listener will be able to open. Also, the OSDI listener JCL procedure name must have an associated z/OS user ID in order to use TCP/IP, which is controlled by z/OS UNIX System Services. The user ID must have an OMVS RACF segment (or equivalent, if a product other than RACF is used) if the installation is not using a default OMVS segment.

In addition, the interface resolves names through the standard `GETHOSTBYNAME` API. Thus the resolution depends on how IBM TCP/IP is configured. If a DNS is defined to TCP/IP, then it will be used. Otherwise, TCP/IP will default the processing to its `SITEINFO` file. Also, IBM's Language Environment run-time library (LE) must be available through a `STEPLIB` DD or linklist to the OSDI listener address space in order for `GETHOSTBYNAME` to work. This is an IBM requirement. TNS does a `GETHOSTBYNAME` call at startup to test the function. This call may take minutes to complete if a busy name server is involved. The interface is not ready for work until the MIN0713I message is displayed on the system console. For more information about the `GETHOSTBYNAME` API, refer to the relevant IBM documentation on TCP/IP.

Client-Server Access Using the OSDI Listener

Note: In this section, the term 'client' refers to the Oracle integrating server in the context of a database link session to an OSDI gateway instance.

Remote Clients

Remote (inbound) clients access Oracle gateway instances through the OSDI listener as follows:

1. The OSDI network service listens on a single endpoint (network address) for each protocol. All remote clients that go through a particular OSDI listener with a particular protocol use the same network address regardless of which

gateway instance they want to access. All TCP/IP clients specify the same host name (or IP) and port number.

2. Clients indicate the target gateway instance that they want with the `'(CONNECT_DATA=(SID=`*`ssss`*`))'` clause in the OSDI listener address string.

   The following is an example of a tnsname entry for a database link from a remote Oracle database server to an OSDI gateway instance through TCP/IP.

```
(DESCRIPTION=
   (ADDRESS= (PROTOCOL=TCP)(HOST=MVS08)(PORT=1999)  )
   (HS=)
   (CONNECT_DATA=(SID=TGD4) )
  )
```

   MVS08 is the host name that the gateway resides in. 1999 is the port number that OSDI NET listens on that serves the gateway instance. TGD4 is the OSDI TG4DB2 SID it is trying to connect to.

   Note that if the remote Oracle database server is an OSDI database instance of release 9.1 or earlier, then you would need to specify SSN parameter (within the ADDRESS specification) that specifies the OSDI NET SID that serves the outbound traffic for the OSDI database instance.

   For MPM and OSDI compatibility, refer to Chapter 11.

Oracle clients on z/OS are also able to use an Oracle Names or LDAP server running on another platform to resolve connection requests. The following samples of the OSDI listener configuration file are required to make use of this service.

Name Server

`SQLNET` DD or `SQLNET.ORA` Definitions

```
###############
# Names .........:  (CONNECT_TIMEOUT = 0) -MUST- be specified
###############
NAMES.DEFAULT_DOMAIN = world
NAMES.DEFAULT_ZONE = my.domain.com
NAMES.DIRECTORY_PATH = (TNSNAMES,ONAMES,LDAP)
NAMES.PREFERRED_SERVERS =
     (ADDRESS_LIST =
       (DESCRIPTION =
         (ADDRESS =
           (PROTOCOL = TCP)
```

```
            (HOST = names_host)
            (Port = 1575)
          )
          (CONNECT_TIMEOUT = 0)
        )
      )
##
```

LDAP Server

`LDAP` DD or `LDAP.ORA` Definitions

A sample `LDAP.ORA` file:

```
DEFAULT_ADMIN_CONTEXT = "c=us"
DIRECTORY_SERVERS = (hostname:389:636)
DIRECTORY_SERVER_TYPE = OID
```

`LDAP.ORA` can be generated using the NETCA utility.

Operating the OSDI Listener

The OSDI listener is started by the OSDI subsystem start command, for example:

```
ORSS START NET
```

This command would start the OSDI listener defined in the earlier example for "Example of Network Service Definition" on page 6-6 if the subsystem were named 'ORSS'. You should then see the OSDI listener PROC start up followed by the following messages from the OSDI listener address space:

```
MIN0001I networking service initializing
MIN0002I networking service NET8     initialization complete
MIN0713I I am listening on port 01521 socket 00000
```

Additional messages are written to the `NET8LOG` DD, but message traffic to the console is limited to error and warning messages.

Several commands are available for communicating with a running Net service. Commands are issued using the z/OS `MODIFY` (or `F`) system operator command with the general format:

`F` *`name,cccc pppppp`*
where:

Variable	Description
name	is the jobname or identifier of the OSDI listener
cccc	is a command verb from the table below
pppppp	represents an appropriate parameter for that command

Table 6–1 Command Verbs for z/OS MODIFY (or F) System Operator Command

Command	Parameter	Description
`start`	`hpns`	Starts support for the specified protocol in the OSDI listener.
`stop`	`hpns`	Stops support for the specified protocol.
`dis`	`tcp | all | pool`	Displays information about existing connections for the specified protocol or storage pool statistics.

The OSDI listener can be stopped with the z/OS stop command (`STOP` or `P`), as in '`p net`', or through the OSDI subsystem stop command, as in '`ORSS STOP NET`'. In either case, the following messages will be seen on the console:

```
MIN0098I networking service NET termination in progress
MIN0721I HPNS shut down, GoodBye.
MIN0099I networking service termination complete
```

The OSDI listener service will also respond to the OSDI subsystem '`display`' and '`display long`' commands with appropriate information from the address space. Finally, the OSDI subsystem '`drain`' command will prevent any new connections on either protocol. Existing connections will not be affected. The OSDI subsystem '`RESUME`' command will restore the ability of clients to establish new connections through the OSDI listener.

Formatting OSDI Listener Trace Files

The OSDI listener provides a utility program called `TRCASST` that formats the trace files OSDI listener can produce. You may be asked to run `TRCASST` to help gather diagnostic information required by Oracle Support Services. Sample JCL for `TRCASST` is provided in `oran.orav.SRCLIB(TRCASST)`.

Before you use `TRCASST`, ensure that the trace files have not been created with carriage control. `TRCASST` will be unable to process such files.

When TRCASST runs, the TNSUSMSG DD name must point to a PDS containing a TNSUS message file. This file was placed into oran.orav.MESG(TNSUS) during OSDI listener installation.

Oracle Advanced Security Option Encryption

The OSDI listener supports CHECKSUM and encryption algorithms. The following sections describe a basic method of verifying this feature, if it is to be used by your site. The easiest way to tell if Oracle Advanced Security Option (ASO) encryption is attempting to work is to deliberately set wrong configuration parameters and attempt a connection between the server and client. Incorrect parameters cause the connection to fail.

After receiving the expected failure message, set the configuration parameters to the correct settings and try the connection again. ASO encryption is working properly if no further error messages are received.

The following procedures test ASO encryption by this method. The incorrect parameter settings produce error 12660.

Setting Up ASO Encryption for Test

Perform the following steps to set up ASO encryption:

Checklist for Setting Up ASO Encryption

1. Set ASO encryption parameters for the server.
2. Set ASO encryption parameters for the client.

Step 1: Set ASO Encryption Parameters for the Server

Use ISPF to edit the OSDI listener configuration file on the z/OS system (server system) to add the following parameters and values. If the server is remote (not z/OS), then use the appropriate editor for the server platform to change SQLNET.ORA.

```
SQLNET.CRYPTO_CHECKSUM_SERVER = REJECTED
SQLNET.ENCRYPTION_SERVER = REJECTED
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (MD5)
SQLNET.ENCRYPTION_TYPES_SERVER = (DES40,RC4_40)
SQLNET.CRYPTO_SEED = "abcdefg"
```

The value shown for SQLNET.CRYPTO_SEED is only an example. Set it to the value you want. Refer to the *Oracle Database Advanced Security Administrator's Guide* for more information.

Step 2: Set ASO Encryption Parameters for the Client

Edit the OSDI listener configuration file on the client system to add the following parameters:

```
SQLNET.CRYPTO_CHECKSUM_CLIENT = REQUIRED
SQLNET.ENCRYPTION_CLIENT = REQUIRED
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (MD5)
SLQNET.ENCRYPTION_TYPES_CLIENT = (DES40,RC4_40)
SQLNET.CRYPTO_SEED = "abcdefg"
```

The value shown for SQLNET.CRYPTO_SEED is only an example. Set it to the same value used on the server system.

Testing ASO Encryption

After completing Steps 1 and 2 of the configuration procedure, you are ready to test the operation of the ASO encryption.

Checklist for Testing ASO Encryption

1. Connect client and server.
2. Reset configuration parameters on server.

Step 1: Connect Client and Server

Attempt a connection between the server and client systems. You should receive the following error message:

```
ORA-12660: Encryption or crypto-checksumming parameters incompatible
```

Step 2: Reset Configuration Parameters on Server

Change the ASO encryption parameters on the server to:

```
SQLNET.CRYPTO_CHECKSUM_SERVER = REQUIRED
SQLNET.ENCRYPTION_SERVER = REQUIRED
```

Attempt the connection between the client and server again. If no error message is returned and the connection completes, then ASO encryption is working properly.

7

Administering the Gateway

This chapter describes the basic administration tasks for Oracle Transparent Gateway for DB2, including implementing security strategies and enabling diagnosis and error reporting. Refer to Appendix A, "OSDI Subsystem Command Reference", for the subsystem administration tasks requiring OSDI commands and parameters.

This chapter includes the following sections:

Operation of the Gateway Subsystem with OSDI

Starting and stopping the gateway service is accomplished by using OSDI operating commands. They are normally issued by using a z/OS system operator command interface. The target subsystem is specified by using the command prefix associated with the subsystem, or the subsystem name. All the operating commands take a gateway service name as the first positional parameter. This service name must be the name of a defined service.

START

The `START` command initiates execution of an address space for a specified service. For a gateway service, the service must not already be running.

The command structure for starting a gateway service is shown in the following example:

```
/ssn START name [ PARM( string ) ]
```

Refer to Appendix A, "OSDI Subsystem Command Reference" for the complete syntax of the `START` command.

STOP

The `STOP` command requests termination of a running service. The normal mode of stop changes the service state to stopping and posts the stop request to the service; it is up to the service to comply, presumably after allowing current requests to complete and performing the required cleanup. A force option causes the service address spaces to be terminated involuntarily. The force form of stop requires that a normal stop be issued first. This command has no effect when the service is not running.

The command structure for stopping a service is shown in the following example:

```
/ssn STOP name [ FORCE ]
```

Refer to Appendix A, "OSDI Subsystem Command Reference" for the complete syntax of the `STOP` command.

Controlling Access to OSDI Subsystem Commands

OSDI subsystem command processing includes an authorization check to confirm that the console or the user is allowed to issue the command. You control access to commands by defining resource profiles to the security subsystem and then by granting access (for specific consoles and users) to those resources. If you do not define resource profiles, then the authorization check returns a "resource unknown" indication to OSDI, and OSDI then allows the command to be processed. Thus, the default behavior (in the absence of any profile definitions) is that any command is allowed from any source. This is not chaotic, as it may sound, because access to command-issuing mechanisms themselves (such as consoles) is usually controlled in most z/OS installations. Base your decision to define profiles and to activate the authorization mechanism upon the security standards and procedures at the installation.

The resource profiles that are used to protect commands should be defined in the resource class that you specified in the cmd-class field of the `INIT` record in the subsystem parameter file. If you elected to accept the default, then this will be the `FACILITY` class. Otherwise, it will be a class name that you chose and configured for your SAF-compliant security software.

The command authorization resource names are of the form:

ssn.cmdverb

where ssn is the OSDI subsystem name, and cmdverb is the full-length OSDI command verb. (Verb abbreviations, such as DEF and ALT, must not be used in the resource name.)

The level of authorization that must be granted to users or to consoles in order to enable commands depends on the command. The following table lists all of the command verbs and the authorization level required for each:

Table 7–1 Command Verbs and their Authorization Level

Verb	Authorization Level
`DEFINE`	Update
`ALTER`	Control
`SHOW`	Read
`START`	Read
`DISPLAY`	Read
`DRAIN`	Read
`RESUME`	Read
`STOP`	Read

Controlling Access to OSDI Services

OSDI bind processing, which establishes connections between z/OS address spaces, performs an authorization check to confirm that the binding address space (the "client") is allowed to access the target service.

In the case of Oracle gateways, the only potential sensible case is that "the client" is the OSDI integrating database server and the target service is the Oracle gateway.

In order for the check for a given target service to be meaningful, resource profiles must be defined to a SAF-compliant security server, such as RACF. The profile

names incorporate the OSDI service name so that access to each service is separately controlled. When profiles are defined, the z/OS user ID that is associated with the client address space must have READ authorization on the target service's profile in order for the bind to be allowed.

If you do not define resource profiles for a service, then all binds from all address spaces are permitted. Oracle Corporation recommends that you define resource profiles for all services so that bind access is controlled through standard z/OS security mechanisms.

The resource profiles that are used to protect binds should be defined in the resource class that you specified in the bind-class field of the INIT record in the subsystem parameter file. If you elected to accept the default, then this will be the FACILITY class. Otherwise, it will be a class name that you chose and configured for your SAF-compliant security software.

The name structure for the managed binds used by Oracle database links is:

ssn.service.BIND

where ssn is the OSDI subsystem name, service is the target service name (which is the gateway), and BIND is a constant indicating managed binds.

Gateway Security

The Oracle user ID and password are passed over the database link to the gateway to authorize gateway users to DB2 objects. If the CONNECT TO clause is specified when creating the database link, then the user ID and password sent to the gateway are those specified in this clause. If the CONNECT TO clause is omitted from the database link specification, then the Oracle user ID and password using the database link are passed to the gateway for authorization.

When the Oracle gateway receives the user ID and password from the database link, a security environment is established that is local to the z/OS and DB2 systems. The following steps occur when establishing a gateway security environment:

1. A gateway security user exit is called by the gateway to verify that the database link user ID and password are valid for z/OS and to set up the security environment for the gateway. The sample user exit delivered with the gateway is called G4RRSAF. One of the tasks of G4RRSAF is to create an ACEE for the current TCB that contains the DB2 primary and secondary authorization ids. This is used by the DB2 RRSAF signon processing.
2. The gateway initiates a connection to DB2 by using a DB2 RRSAF SIGNON.

The remaining sections of this chapter provide more specific information about gateway security, including:

- z/OS Security Authorization Facility (SAF) considerations
- Gateway User Exit Facility

SAF Router Considerations

The gateway uses the SAF router to validate userids and passwords. It is also used to check user access to the DB2 security profile, with `CLASS` set to `DSNR` and `ENTITY` set to `db2_subsys.BATCH`. The DB2 security profile is optional.

Security environments using the SAF interface include `ACF2`, `RACF`, and `TOP SECRET`. If your local security environment does not use the SAF interface, then your system programmer can replace the SAF calls where appropriate.

Gateway User Exit Facility

The gateway user exit facility provides gateway installations with a mechanism for passing control to installation-written code at user logon time.

The user exit module is not part of the gateway or OSDI. It is a z/OS load module (exit module). During OSDI startup, these user exit modules are loaded by OSDI dynamically into the gateway address space from one of these locations:

- The OSDI `STEPLIB`
- The system linklist (`LNKLST00`)

Specifying an Exit Module

Exit modules are specified to the gateway with the gateway region parameter, `LOGON_AUTH`, which is accessed at OSDI initialization time:

```
LOGON_AUTH(name)
```

where `name` is the exit module member name. `G4RRSAF` is supplied to provide the standard SAF security check.

To display the status of an `EXIT`, use the OSDI `DISPLAY EXIT` command.

The logon exit must reside in the `STEPLIB` or `JOBLIB` concatenation or in the linklist, and it must be in an authorized library.

A sample user logon exit is provided in *oracle_hlq*.SRCLIB library member G4SIGNON. It uses the z/OS SAF interface to invoke the z/OS security manager. If the z/OS security manager does not support the SAF interface, then the calls to RACROUTE in the exit must be replaced with the equivalent calls appropriate for the z/OS security manager.

The calling sequence for the logon exit uses standard z/OS assembler calling conventions. R15 is the entry point, R14 is the return address, R13 points to a standard 72-byte save area, and R1 is the address of a parameter list. The parameter list consists of a list of addresses of each parameter (all values are passed by reference, not by value), and the last parameter has the high-order bit set.

When returning, R15 should be set to 0 to indicate a successful verification of the user ID and password that were supplied, and should be set to any nonzero value to indicate any type of failure (4 would be an appropriate value).

The exit is called in 31-bit addressing mode, supervisor state, storage protection key 7, and in an authorized address space. The exit will be running in TCB mode with no locks held and with no ARRs, FRRs, or EUT-style FRRs set. The exit is called in primary addressing mode with HASN=PASN=SASN (home, not cross-memory mode).

The logon exit should be fully reentrant code.

The parameter list that is passed contains pointers to the following parameters, all of which are input only:

Table 7–2 Gateway Region Parameters

Field	Type/Length	Description
work area	char/4k	set to all x'00' before every call to the exit
userid	char/1+	user ID to be validated (number of bytes varies)
userid length	bin/2	length of user ID
password	char/1+	password to be validated (number of bytes varies)
password length	bin/2	length of password
z/OS jobname	char/8	z/OS jobname from JOB card of client
ASID	bin/2	address space id of client address space
OSDI session id	bin/4	a unique OSDI session id
OS username	char/8	the operating system user name (batch, TSO, CICS, and IMS only)

Table 7–2 (Cont.) Gateway Region Parameters

Field	Type/Length	Description
terminal name	char/8	terminal name
program name	char/8	program name
RACF group name	char/8	RACF group name
connection type	char/8	connection type (BATCH, TSO, CICS, IMS, TCP/IP, VTAM) to indicate environment of client
JES jobid	char/8	JES job identifier (such as JOB08237)
job card entry time	bin/4	entry (submission) time of job. Binary hundredths of a second since midnight
job card entry date	packed/4	entry (submission) date of job. Packed decimal 0CYYDDDF, where C-0 for 19, C=1 for 20, YY=year, DDD=day number within the year (Jan 1=1)
job card accounting info	char/145	from jobcard
network data (high bit set in parameter list)	char/2+	variable length NIV data (refer to Chapter 9, "Oracle SMF Data")

The only output of the logon exit is the R15 return code. No other value that is passed in the parameter list should be modified except the first one.

The first parameter is a 4096-byte work area that is set to all x'00' before every call to the logon exit. The logon exit can use this storage for anything that it needs. It should not be freed.

The logon exit can do any of the following:

- call a security manager (SAF calls, RACF, Top Secret, ACF2, and so forth)
- get and free storage using the STORAGE macro (the exit must keep track of all acquired storage and must make certain to free it before exiting)
- call SMF to write SMF records
- call WTO to write messages to the console

The logon exit should not do anything that would cause it to wait for any significant period of time (more than one-tenth of a second, for example). Avoid

opening data sets, writing to the operator with a reply (WTOR), and creating enqueues.

Any resource that is acquired in the logon exit must be freed before it returns. There is no cleanup call made to the logon exit, so any resources that are not released will accumulate in the address space and could eventually cause resource shortages.

Sample Exit Programs

Sample Exit Programs are supplied in the *oracle_hlq*.SRCLIB data set.

The DB2 date exit is called DSNXVDTX. The sample JCL to assemble and link it is called XVDTXCL.

The DB2 RRSAF logon security exit is shipped as an executable called G4RRSAF in the AUTHLOAD library. If you need to customize the exit for the installation, then the source needed to build G4RRSAF is called G4SIGNON in the *oracle_hlq*.SRCLIB data set. The JCL to assemble and link G4SIGNON is in SIGNONCL.

8

Using the Gateway

Using the gateway involves connecting to the gateway system and the remote DB2 database associated with it.

This chapter includes the following sections:

Database Link Behavior

A connection to the gateway is established through a database link when it is first used in a gateway session or transaction. In this context, connection refers to the connection between the Oracle database server and the gateway. The connection remains established until the session ends. Another session or user can access the same database link and get a connection to the gateway and DB2 database.

Connections to the DB2 database might be limited by factors that include memory, gateway parameters, or DB2 server resources.

The database and application administrators of a distributed database system are responsible for managing the necessary database links defining paths to the gateway.

Database links are discussed in detail in the *Oracle Database Administrator's Guide*. Information for using database links with the gateway is discussed here.

Creating Database Links

To create a database link and to define a path to the gateway, use the `CREATE DATABASE LINK` statement. The `CONNECT TO` clause specifies the remote user ID and password to use when creating a session in the gateway. If you do not specify a user ID and password in the `CONNECT TO` clause, then the Oracle logon user ID and password are used. The `USING` clause specifies a `TNSNAMES.ORA` connect descriptor.

Creating Database Links Using Oracle Net

The following syntax creates a database link to access information in the DB2 database using Oracle Net:

```
CREATE DATABASE LINK dblink
       CONNECT TO userid identified by password
       USING 'tns_name_entry';
```

where:

Parameter	Description
`dblink`	is the complete database link name (such as gateway).
`userid`	is the user ID used to establish a session in the remote database. It must be authorized to any table on the DB2 server referenced in the SQL commands. The user ID cannot be longer than eight characters.
`password`	is the password used to establish a session in the remote database. This must be a valid DB2 server password. The password cannot be longer than eight characters.
`tns_name_entry`	specifies the Oracle Net TNS connect descriptor used to identify the gateway instance. For an example of a tnsnames entry, refer to Chapter 6.

Guidelines for Database Links

Once used, a database link remains open for the duration of the gateway session. If you want to close a database link during a session, then you can do so with the `ALTER SESSION CLOSE DATABASE LINK` statement.

Accessing Data through Database Links

DB2 tables, views, synonyms, and aliases available to the userid specified in the `CONNECT TO` clause can be accessed with the following syntax:

```
SELECT * FROM SCOTT.EMP@gateway
```

The `CONNECT TO` userid provides implicit qualification for unqualified tables. For example:

```
SELECT * FROM EMP@gateway
```

resolves to `SCOTT.EMP` on DB2 if the `CONNECT TO` user is `SCOTT`. If no `CONNECT TO` statement is defined with the database link, then the Oracle user ID using the database link is used as the implicit qualifier.

Dropping Database Links

You can drop a database link with the `DROP DATABASE LINK` statement. For example, to drop the public database link named `dblink`, enter the following statement:

```
DROP DATABASE LINK dblink;
```

Do not drop a database link if it is required to resolve an in-doubt distributed transaction. Refer to the *Oracle Database Administrator's Guide* for additional information about dropping database links.

Examining Available Database Links

The data dictionary of each database stores the definitions of all the database links in that database. The `USER_DB_LINKS` data dictionary view shows the database links defined for a specific Oracle user. The `ALL_DB_LINKS` data dictionary views show all defined database links both public and private. The user has access to all these views. The `DBA_DB_LINKS` dictionary view, accessible only to users with DBA authorization, shows all database links defined in the gateway instance.

Limiting the Number of Active Database Links

You can limit the number of connections from a user process to remote databases with the INIT.ORA parameter OPEN_LINKS. This parameter controls the number of remote connections any single user process can use concurrently with a single SQL statement. Refer to the *Oracle Database Administrator's Guide* for additional information about limiting the number of active database links.

Managing Threads

Whenever a client connects to the Oracle database server to access data from a DB2 server, the Oracle database server creates and manages database access threads between the client and the Oracle database server. DB2 creates and manages allied threads between the gateway and DB2. The Oracle system manages the threads between the client and the Oracle database server. The DB2 system manages the threads between the gateway and DB2.

When using the gateway to access DB2 data in a client/server configuration, you can encounter the following scenarios:

1. A user turns off a workstation abnormally and the thread connection remains active.
2. A user leaves a workstation in an idle state for an extended period of time and the thread connection remains active.
3. A user or application uses a workstation to enter a long-running query that maintains a lock in DB2 and, thus, an active thread.

When connecting to DB2 through the gateway, it is important to remember that, if the client or server is abnormally terminated, then a connection can be left open indefinitely, unless specifically identified and closed by the system.

KEEPALIVE

The KEEPALIVE functionality can be used to resolve the first of the three scenarios mentioned previously. For Oracle systems using KEEPALIVE, the identification of an inactive client/server connection is handled differently in the UNIX environment from the way it is handled in the z/OS environment.

With UNIX, an optional parameter SQLNET.EXPIRE_TIME in the SQLNET.ORA file determines how often Oracle Net sends a probe to verify whether a client/server connection is still active. If the connection is inactive, then the Oracle database

server cleans up the connections between the client, the Oracle database server, the gateway, and DB2.

With z/OS, KEEPALIVE sends a probe to verify whether a client/server connection is still active. The KEEPALIVE functionality is implicitly leveraged by the individual protocol vendors. For example, if you are using the TCP/IP protocol and KEEPALIVE is enabled, then the KEEPALIVE functionality is used automatically by Oracle Net for z/OS.

Canceling DB2 Threads

The DB2 command CANCEL THREAD can potentially be used to alleviate problems in scenario three by scheduling threads to be terminated. The user or application must still attempt to access DB2 again before the thread can be terminated.

For further information about this feature, refer to the IBM documents for your platform and operating system.

Gateway CPU Time

Oracle Transparent Gateway for DB2 uses the RRSAF to connect to the target DB2 system. When the gateway is used to access data in DB2, much of the DB2 SQL processing takes place in cross-memory mode, running under a TCB in the gateway address space with the RRSAF. In turn, the CPU time charged to the gateway address space includes both CPU utilization of the gateway and the CPU time required for DB2 SQL processing. Thus, the SMFXMCPU field, which records CPU utilization, is a combination of the gateway and the DB2 CPU time. Therefore, this field does not represent pure gateway CPU time and is not a good predictor for judging gateway efficiency.

Using DB2 Cursors

The maximum number of DB2 cursors the gateway can open per Oracle session is 5000. Although the gateway can open 5000 cursors, other Oracle database server or DB2 limits might affect how many cursors can actually be opened for a specific application. The default is 50.

Ensure that the HS_OPEN_CURSORS parameter in member G4DB2ENV of the PARMLIB library is set to the maximum you require.

Using the Synonym Feature

You can provide complete data, location, and network transparency by using the synonym feature of the Oracle database server. When a synonym is defined, you do not need to know the underlying table or network protocol being used. A synonym can be public, which means all users can make reference to the synonym. A synonym can also be defined as private, which means every user must have a synonym defined to access a DB2 table. Refer to the Oracle Database product documentation for details about the synonym feature.

The following statement creates a system wide synonym `EMPDB2` for the EMP file in the DB2 server Oracle library:

```
CREATE PUBLIC SYNONYM EMPDB2 FOR SCOTT.EMP@gateway
```

Only those with database administrator authority can create public synonyms. You can use a similar statement to create a private synonym if you do not have database administrator authority:

```
CREATE SYNONYM EMPDB2 FOR SCOTT.EMP@gateway
```

Read-Only Gateway

The read-only option can provide improved performance and security based on your configuration and parameter selections. An environment parameter, `DB2READONLY`, is used to control whether the gateway is enabled in this mode.

If you enable the read-only feature, then only queries (`SELECT` statements) are allowed by DB2. The capabilities that control whether updates are allowed through the gateway are disabled. These capabilities include `INSERT`, `UPDATE`, `DELETE`, and stored-procedure support (pass-through SQL and DB2 stored procedures). Statements attempting to modify records in DB2 are rejected.

Oracle Corporation recommends that you do not routinely switch between settings of the `DB2READONLY` parameter. If you need both update and `DB2READONLY` functionality, then you should install two separate instances of the gateway with different read-only settings.

If your system can tolerate an occasional dirty read, then you can bind the gateway plan using the isolation level (uncommitted read). This eliminates DB2 locking problems and improves overall performance.

Performing Distributed Queries

The gateway technology enables the completion of distributed queries joining data from the Oracle database server and the DB2 server, and any other data store for which Oracle provides a gateway. These complex operations are transparent to the users requesting such data retrieval.

Example of a Distributed Query

The following example joins data between the Oracle database server and multiple DB2 servers:

```
SELECT O.CUSTNAME, P.PROJNO, E.ENAME, SUM(E.RATE*P.HOURS)
FROM ORDERS@GATEWAY_1 O, EMP@ORACLE9 E, PROJECTS@GATEWAY_2 P
WHERE O.PROJNO = P.PROJNO
AND P.EMPNO = E.EMPNO
GROUP BY O.CUSTNAME, P.PROJNO, E.ENAME;
```

Through a combination of views and synonyms, using the following SQL statements, the process of distributed queries is made transparent to the user:

```
CREATE SYNONYM ORDERS FOR ORDERS@GATEWAY_1;
CREATE SYNONYM PROJECTS FOR PROJECTS@GTEWAY_2;
CREATE VIEW DETAILS (CUSTNAME,PROJNO,ENAME,SPEND)
  AS
  SELECT O.CUSTNAME, P.PROJNO, E.ENAME, SUM(E.RATE*P,HOURS)
  FROM ORDERS O, EMP E, PROJECTS P
  WHERE O.PROJNO = P.PROJNO
  AND P.EMPNO = E.EMPNO
     GROUP BY O.CUSTNAME, P.PROJNO;
```

With the views and synonyms in place, the user retrieves information from these three data stores by using one command:

```
SELECT * FROM DETAILS;
```

which produces the following:

CUSTNAME	PROJNO	ENAME	SPEND
ABC Co	1	Jones	400
ABC Co.	1	Smith	180
XYZ Inc.	2	Jones	400

CUSTNAME	PROJNO	ENAME	SPEND
XYZ Inc.	2	Smith	180

Two-Phase Commit Processing

The gateway must coordinate the distributed transaction and only one gateway can participate in an Oracle two-phase commit transaction.

Two-phase commit transactions are recorded in the DB2 table `ORACLE2PC`, which is created during gateway installation.

On all systems, the `ORACLE2PC` table must be available at all times. For security reasons, users must not have direct access to this table. The table is accessed and updated by the gateway internally.

Distributed DB2 Transactions

Because the `ORACLE2PC` table is used to record the status of a gateway transaction, the table must reside in the database where the DB2 update takes place. Updates to the `ORACLE2PC` table cannot be part of an IBM distributed transaction.

Replicating in a Heterogeneous Environment

Oracle Transparent Gateway for DB2 provides a number of options for replicating Oracle and non-Oracle data throughout the enterprise.

Oracle Database Server Triggers

When updates are made to the Oracle database server, synchronous copies of Oracle and non-Oracle data can be maintained automatically by using Oracle database server triggers.

Oracle Materialized View

Oracle Transparent Gateway for DB2 can use the Oracle materialized view feature to automatically replicate non-Oracle data into the Oracle database server. This complete refresh capability of Oracle materialized view can be used to propagate a complete copy or a subset of the non-Oracle data into the Oracle database server at user-defined intervals.

Copying Data from the Oracle Database Server to the DB2 Server

Data can be copied from the Oracle database server to the DB2 server by two methods:

- Triggers
- SQL*Plus COPY command

Triggers

When updates are made to the Oracle database server, synchronous copies of Oracle and non-Oracle data can be maintained automatically by using Oracle database server triggers.

For example, you have an Oracle ORA_EMP table that contains ENAME and EMPNO. You also have a table called DB2_EMP, which is a copy of ORA_EMP and which resides on DB2. You want all changes made to the Oracle ENAME to be reflected immediately in the DB2_EMP table on DB2. In this scenario, an Oracle database server trigger can be developed to run every time an update is made to ENAME in your Oracle ORA_EMP table:

```
CREATE OR REPLACE TRIGGER EMP_TRIGGER
 AFTER UPDATE OF ENAME ON SCOTT.ORA_EMP
 FOR EACH ROW
BEGIN
     UPDATE SCOTT.DB2_EMP@tg4db2
     SET ENAME = :NEW.ENAME
     WHERE EMPNO = :NEW.EMPNO;
END;
```

where *tg4db2* is the name of the database link used to access the gateway.

SQL*Plus COPY Command

The SQL*Plus COPY command copies data from the Oracle database server to the DB2 server. The SQL command INSERT is not supported. The command:

```
INSERT INTO gateway_table SELECT * FROM oracle_table;
```

displays the following message:

```
ORA-2025: All tables in the SQL statement must be at the remote
          database.
```

Use the following SQL*Plus syntax to copy data from your local Oracle database server to the DB2 server:

```
COPY FROM username/password@ORACLE9
INSERT destination_table
USING query;
```

The next example selects all rows from the local Oracle EMP table and inserts them into the EMP table on the DB2 server:

```
COPY FROM SCOTT/TIGER@ORACLE9
INSERT SCOTT.EMP@gateway
USING SELECT * FROM EMP;
```

Note: The SQL*Plus COPY command supports APPEND, CREATE, INSERT, and REPLACE options.

However, INSERT is the only option supported when copying to the DB2 server.

For more information about the COPY command, refer to the *SQL*Plus User's Guide and Reference*.

STREAMS Replication

TG4DB2 and Heterogeneous Services now support replication to DB2 using Oracle streams. Oracle Streams is a rule-based process which allows changes to an Oracle table to be captured and applied to an equivalent DB2 table based on user-written rules. Replication from DB2 to Oracle is not supported at this time.

Before setting up your Streams Replication environment ensure that archivelog is enabled, otherwise nothing else will work.

An example of a simple table replication follows:

```
First you should grant the necessary authorizations to your Streams admin
userid.
   CONNECT SYS/SYS_PASSWORD AS SYSDBA

   GRANT CONNECT, RESOURCE, SELECT_CATALOG_ROLE
     TO strmadmin IDENTIFIED BY strmadminpw;

   GRANT EXECUTE ON DBMS_APPLY_ADM TO strmadmin;
   GRANT EXECUTE ON DBMS_AQADM TO strmadmin;
   GRANT EXECUTE ON DBMS_CAPTURE_ADM TO strmadmin;
```

```
GRANT EXECUTE ON DBMS_FLASHBACK TO strmadmin;
GRANT EXECUTE ON DBMS_PROPAGATION_ADM TO strmadmin;
GRANT EXECUTE ON DBMS_STREAMS_ADM TO strmadmin;

BEGIN
  DBMS_RULE_ADM.GRANT_SYSTEM_PRIVILEGE(
  privilege => DBMS_RULE_ADM.CREATE_RULE_SET_OBJ,
  grantee => 'strmadmin',
  grant_option => FALSE);
END;
/

BEGIN
  DBMS_RULE_ADM.GRANT_SYSTEM_PRIVILEGE(
  privilege => DBMS_RULE_ADM.CREATE_RULE_OBJ,
  grantee => 'strmadmin',
  grant_option => FALSE);
END;
/
```

Then, set up the Streams queue and the database link that the apply process will use.

```
CONNECT strmadmin/strmadminpw

EXEC DBMS_STREAMS_ADM.SET_UP_QUEUE();

DROP DATABASE LINK strmdblink.your.domain.com;

CREATE DATABASE LINK strmdblink.your.domain.com
  CONNECT TO userid IDENTIFIED BY password
     USING 'tnsnames_entry';
```

Next, create the capture and apply processes and define the replication rules.

```
CONNECT SYS/SYS_PASSWORD AS SYSDBA

ALTER SYSTEM ARCHIVE LOG CURRENT;

CONNECT strmadmin/strmadminpw

---  ------------------------------------------------------------
---  Stop the capture process if it's already active.
```

```
---  ------------------------------------------------------------
BEGIN
  DBMS_CAPTURE_ADM.STOP_CAPTURE(
    capture_name => 'db2_capture');
END;
/

---  ------------------------------------------------------------
---  Stop the apply process if it's already active.
---  ------------------------------------------------------------
BEGIN
  DBMS_APPLY_ADM.STOP_APPLY(
    apply_name => 'apply_2_db2');
END;
/
---  ------------------------------------------------------------
---  Define the capture rule, this one captures changes to scott.emp
---  ------------------------------------------------------------
BEGIN
  DBMS_STREAMS_ADM.ADD_SCHEMA_RULES(
    schema_name  => 'scott',
    streams_type => 'capture',
    streams_name => 'db2_capture',
    queue_name   => 'strmadmin.streams_queue',
    include_dml  => true,
    include_ddl  => true);
END;
/

---  ------------------------------------------------------------
---  Set the capture instantiation level
---  ------------------------------------------------------------
DECLARE
  iscn NUMBER; -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();

  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN(
    source_object_name => 'scott.emp',
    source_database_name => 'ORAv92',
    instantiation_scn => iscn,
    apply_database_link => 'strmdblink.your.domain.com');
END;
/
```

```
---  ------------------------------------------------------------
---  Drop the apply process if it already exists.
---  ------------------------------------------------------------
BEGIN
  DBMS_APPLY_ADM.DROP_APPLY(
    apply_name => 'apply_2_db2');
END;
/

---  ------------------------------------------------------------
---  Create the apply process
---  ------------------------------------------------------------
BEGIN
  DBMS_APPLY_ADM.CREATE_APPLY(
    queue_name => 'strmadmin.streams_queue',
    apply_name => 'apply_2_db2',
    apply_database_link => 'strmdblink.your.domain.com',
    apply_captured => true);
END;
/

---  ------------------------------------------------------------
---  Create the apply rule
---  ------------------------------------------------------------
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name => 'scott.emp',
    streams_type => 'apply',
    streams_name => 'apply_2_db2',
    queue_name => 'strmadmin.streams_queue',
    include_dml => true,
    include_ddl => false,
    source_database => 'ORAv92');
END;
/

---  ------------------------------------------------------------
---  Turn on tracing for the apply process (be careful, this
---  generates alot of output).
---  ------------------------------------------------------------
BEGIN
  DBMS_APPLY_ADM.SET_PARAMETER(
    apply_name => 'apply_2_db2',
    parameter => 'trace_level',
    value => 127 );
```

```
END;
/
---  ------------------------------------------------------------
---  Turn off disable_on_error for the apply process
---  ------------------------------------------------------------
BEGIN
  DBMS_APPLY_ADM.SET_PARAMETER(
    apply_name => 'apply_2_db2',
    parameter => 'disable_on_error',
    value => 'n');
END;
/

---  ------------------------------------------------------------
---  Start the apply process.
---  ------------------------------------------------------------
BEGIN
  DBMS_APPLY_ADM.START_APPLY(
    apply_name => 'apply_2_db2');
END;
/

---  ------------------------------------------------------------
---  Start the capture process.
---  ------------------------------------------------------------
BEGIN
  DBMS_CAPTURE_ADM.START_CAPTURE(
    capture_name => 'db2_capture');
END;
/
```

For more detailed information about Oracle streams replication, refer to *Oracle Streams Concepts and Administration*.

Copying Data to the Oracle Database Server from the DB2 Server

Use one of the following options to copy data from the DB2 server to the Oracle database server:

- Use the `CREATE TABLE` command to copy data from the DB2 server to the Oracle database server. To create a table on your local database and insert rows from a DB2 table, use:

  ```
  CREATE TABLE table_name AS query;
  ```

The next example creates the table EMP in the local Oracle database server and inserts the rows from the EMP table on the DB2 server:

```
CREATE TABLE EMP AS SELECT * FROM SCOTT.EMP@gateway;
```

- Use the INSERT command to copy data from the DB2 server to the Oracle database server:

  ```
  INSERT INTO oracle_table SELECT * FROM db2table@gateway;
  ```

  The following example selects all rows from the EMP table on the DB2 server and inserts them into the local Oracle EMP table:

  ```
  INSERT INTO EMP SELECT * FROM SCOTT.EMP@gateway;
  ```

- Use the CREATE MATERIALIZED VIEW command to automatically and asynchronously copy DB2 server data into the Oracle database server. The complete refresh capability can be used to propagate a complete copy or a subset. For more information about creating MATERIALIZED VIEWs, refer to the *Oracle Database SQL Reference*. To create a copy:

  ```
  CREATE MATERIALIZED VIEW empdb2
    PCTFREE 5 PCTUSED 60
    TABLESPACE users
    STORAGE (INITIAL 50K NEXT 50K)
    REFRESH COMPLETE NEXT SYSDATE + 1
    WITH ROWID
    AS
      SELECT * FROM SCOTT.EMP@gateway;
  ```

  The following example creates a materialized view of data that is refreshed every day after the first refresh. If you require only a subset of the DB2 data, then a WHERE clause is added as in the following example:

  ```
  CREATE MATERIALIZED VIEW empdb2
    PCTFREE 5 PCTUSED 60
    TABLESPACE users
    STORAGE (INITIAL 50K NEXT 50K)
    REFRESH COMPLETE NEXT SYSDATE + 1
    WITH ROWID
    AS
      SELECT * FROM SCOTT.EMP@gateway
    WHERE deptno=20;
  ```

- Use the SQL*Plus COPY command to copy data from the DB2 server to the Oracle database server:

```
COPY FROM username/password@gateway
INSERT destination_table
USING query;
```

The following example selects all rows from the EMP table in DB2 and inserts them into the local Oracle EMP table:

```
COPY FROM SCOTT/TIGER@gateway
INSERT EMP
USING SELECT * FROM SCOTT.EMP@gateway;
```

Note: The Oracle Database 10*g* database server allows only one LONG column per table, which might create a situation where a DB2 table cannot be replicated directly in an Oracle table.

9

Developing Applications

Oracle Transparent Gateway for DB2 allows applications written for the Oracle database server to access tables in a DB2 database. Using a database link, the access can be made transparent by using synonyms or views of the DB2 tables. However, there are fundamental SQL, data type, and semantic differences between the Oracle database server and the DB2 database. Read this chapter to learn these differences and for information about developing applications.

This chapter includes the following sections:

Gateway Appearance to Application Programs

An application written to access information in a DB2 database interfaces with an Oracle database server. When developing applications, remember the following:

- You must define the DB2 database to the application by use of a database link. Your application specifies tables existing on a DB2 database using the name defined in the database link. For example, if you define a database link naming the DB2 database link DB2, and an application needs to retrieve data from an Oracle database server and the DB2 database, then the following SQL statement retrieves data from both Oracle and DB2:

  ```
  SELECT EMP.EMPNO, EMPS.SALARY FROM EMP, EMPS@DB2
       WHERE EMP.EMPNO = EMPS.EMPNO;
  ```

  In this example, `EMP` is a table on the Oracle database server and `EMPS` is a table on the DB2 server. Alternatively, you can define a synonym or a view on the DB2 server table and access the information without the database link suffix.
- You can perform reads and writes of data to a defined DB2 database. `SELECT`, `INSERT`, `UPDATE`, and `DELETE` are all valid operations.
- A single transaction can write to one DB2 database and to multiple Oracle database servers.
- Single SQL statements, using a `JOIN`, can refer to tables in multiple Oracle database servers, multiple DB2 databases, or both.

Array Processing

When evaluating and tuning the gateway configuration, you can achieve performance gains by using the Oracle array processing interface. An array is a collection of data items, called elements, associated with a single variable. With arrays, you can use a single SQL statement to manipulate an entire collection of data items. For example, suppose you want to insert information regarding 100 employees into the `EMP` table on DB2. Without arrays, your program must do 100 individual `INSERT`s—one for each employee. With arrays, only one `INSERT` is necessary.

The use of array processing reduces network calls, which can save elapsed time and CPU cycles. In addition, when using `INSERT` for multiple rows, DB2 processing is optimized by retaining the original SQL statement for repeated running.

You can set the array size between the client and the gateway by using your Oracle application implementation for `UPDATE`, `DELETE`, and `INSERT`.

For more information about array processing usage and implementation in your Oracle application, refer to the *SQL*Plus User's Guide and Reference* or the *Oracle Call Interface Programmer's Guide*.

Note: For performance reasons, Oracle Corporation recommends setting the initial Oracle application array size between 10 and 100.

Fetch Reblocking

The Oracle database server supports fetch reblocking with the HS_RPC_FETCH_REBLOCKING parameter.

When the value of this parameter is set to ON (the default), the array size for SELECT statements is determined by the HS_RPC_FETCH_SIZE value. The HS_RPC_FETCH_SIZE parameter defines the number of bytes sent with each buffer from the gateway to the Oracle database server. The buffer might contain one or more qualified rows from DB2. This feature can provide significant performance enhancements, depending on the application design, installation type, and workload.

The array size between the client and the Oracle database server is still determined by the Oracle application.

Using Oracle Stored Procedures with the Gateway

The gateway stored procedure support is an extension of Oracle stored procedures. An Oracle stored procedure is a schema object that logically groups a set of SQL and other PL/SQL programming language statements together to perform a specific task. Oracle stored procedures are stored in the database for continued use. Applications use standard Oracle PL/SQL to call stored procedures.

Oracle stored procedures can be located in a local instance of the Oracle database server and a remote instance. The following example shows two stored procedures: oraproc1 is a procedure stored in the ORA1 Oracle instance, while oraproc2 is a procedure stored in the ORA2 Oracle instance.

To maintain location transparency in the application, a synonym can be created:

```
CREATE SYNONYM oraproc2 FOR oraproc2@ora2;
```

After this synonym is created, the application no longer needs to use the database link specification to call the stored procedure at the remote Oracle instance.

The second statement in oraproc1 is used to access a table in the ORA2 instance. In the same way, Oracle stored procedures can be used to access DB2 tables through the gateway.

`empproc` is an Oracle stored procedure, which subsequently accesses data in DB2 using the gateway:

Like the Oracle database server, standard PL/SQL is used to create and run the procedure. There is no difference with the gateways except that the stored procedure is accessing DB2 instead of the Oracle database server.

Gateway two-phase commit processing also applies to updates to DB2 being made within an Oracle stored procedure. This means the stored procedure can update a single instance of DB2 while also updating any number of Oracle database servers within a single transaction.

Using DB2 Stored Procedures with the Gateway

The procedural feature of the gateway enables execution of native DB2 stored procedures. In other words, the stored procedure is no longer defined in the Oracle database server, but instead, is defined to DB2. Again, standard Oracle PL/SQL is used by the Oracle application to run the DB2 stored procedure.

The gateway does not require special definitions to call the DB2 stored procedure. Once the stored procedure is defined to DB2, the gateway is able to use the existing DB2 definition to run the procedure.

An Oracle application calls the `empproc` stored procedure that is defined to DB2.

From the perspective of the application, running the DB2 stored procedure is no different than invoking a stored procedure at a remote Oracle instance.

Oracle Application DB2 Stored Procedure Execution

In order for an Oracle application to call a DB2 stored procedure, it is first necessary to create the DB2 stored procedure on the DB2 system using the procedures found in the IBM documents for your platform and operating system.

After the stored procedure is defined to DB2, the gateway is able to access the data using a standard PL/SQL call. For example, an employee name, JOHN SMYTHE, is passed to the DB2 stored procedure REVISE_SALARY. The DB2 stored procedure retrieves the salary value from the DB2 database to calculate a new yearly salary for `JOHN SMYTHE`. The revised salary returned in `RESULT` is used to update the `EMP` table of an Oracle database server:

```
DECLARE
  INPUT VARCHAR2(15);
  RESULT NUMBER(8,2);
BEGIN
```

```
  INPUT := 'JOHN SMYTHE';
  SYSPROC.REVISE_SALARY@DB2(INPUT, RESULT);
  UPDATE EMP SET SAL = RESULT WHERE ENAME = INPUT;
END;
```

When running a DB2 stored procedure, a two-part procedure name is sent to DB2 through the gateway. If no qualifier is used in the Oracle application to call the stored procedure, then the userid passed over the database link, or `PUBLIC`, is used as the qualifier for the procedure name.

DB2 stored procedures migrated from DB2 V5.1 require that `SYSPROC` must be the first qualifier of a stored procedure name. Therefore, the application must ensure `SYSPROC` is used as the qualifier for the DB2 stored procedure. One way to do this is to explicitly qualify the procedure name:

```
BEGIN
  SYSPROC.PROC1(parm1)
END
```

DB2 stored procedures for DB2 V6.1 and subsequent allows stored procedures to be qualified by schema name other than `SYSPROC`.

Procedural Feature Considerations with DB2

The following are special considerations for using the procedural feature with the gateway:

- DB2 stored procedures do not have the ability to coordinate, commit, and rollback activity on recoverable resources such as IMS or CICS transactions. Therefore, if the DB2 stored procedure calls a CICS or IMS transaction, then it is considered a separate unit of work and does not affect the completion of the stored procedure. This means that if you are running a DB2 stored procedure from an Oracle application and this procedure calls a CICS or IMS transaction, then the gateway cannot recover from any activity that occurred within the CICS or IMS transaction.

  For example, the CICS transaction could rollback a unit of work, but this does not prevent the gateway from committing other DB2 work contained within the DB2 stored procedure.

  Likewise, if the DB2 stored procedure updated an irrecoverable resource such as a VSAM file, then the gateway considers this activity separate from its own recoverable unit of work.

- When running DB2 stored procedures containing DB2 SQL, you must have the collection ID of the DB2 package specified in the `CREATE PROCEDURE`. If this was not initially specified, an `ALTER PROCEDURE` can be done to add this.

  This is required because the DB2 plan for the gateway must identify the packages for the DB2 stored procedures it runs. The default DB2 bind JCL, delivered with the gateway, uses `*.*` to identify the package list for the gateway. When this is specified, DB2 identifies the collection ID of the DB2 stored procedure as the one specified in the `COLLID` column of the DB2 stored procedure entry in the `SYSROUTINES` table.
- PL/SQL records cannot be passed as parameters when invoking a DB2 stored procedure.
- The gateway supports `DB2SQL`, `GENERAL` and `GENERAL WITH NULLS` linkage conventions of DB2 stored procedures.
  - The `GENERAL` linkage convention means that the parameters passed to and from the DB2 stored procedure cannot be null.
  - The `DB2SQL` and `GENERAL WITH NULLS` linkage convention means that the parameters passed to and from the DB2 stored procedure can be null when they are passed using indicator variables.
  - Embedded PL/SQL or OCI can be used in the host program to operate on indicator variables. Refer to Appendix E, "Sample Applications", for a sample DB2 stored procedure and PL/SQL program using the `GENERAL WITH NULLS` linkage convention.

Passing DB2 SQL Statements Through the Gateway

The passthrough SQL feature allows an application developer to send a SQL statement directly to DB2 without the statement being interpreted by the Oracle database server. The `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE SQL` passthrough statements supported by the gateway are limited to nonqueries (`INSERT`, `UPDATE`, `DELETE`, and `DDL` statements) and cannot contain bind variables. The gateway can run native DB2 SQL statements by using `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE`.

`DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` is a built-in gateway function. This function receives one input argument and returns the number of rows affected by the SQL statement. For DDL statements, the function returns zero.

`DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` is the reserved name of the gateway and are used specifically for running native DB2 SQL.

This release of Oracle Transparent Gateway for DB2 enables retrieval of result sets from queries issued with passthrough. The syntax is different from the DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE function. Refer to "Retrieving Result Sets Through Passthrough" on page 9-8 for more information.

Using the DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE Function

To run a passthrough SQL statement using DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE, use the following syntax:

```
number_of_rows = DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@dblink ('native_DB2_
sql');
```

where:

Parameter	Description
number_of_rows	is a variable that is assigned the number of rows affected by the passthrough SQL completion. For DDL statements, a zero is returned for the number of rows affected.
dblink	is the name of the database link used to access the gateway.
native_DB2_sql	is a valid DB2 non query SQL statement (except SAVEPOINT, ROLLBACK TO SAVEPOINT, RELEASE TO SAVEPOINT, COMMIT, and ROLLBACK). The statement cannot contain bind variables. DB2 SQL statements that cannot be dynamically prepared are rejected by DB2. The SQL statement passed by the DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE function must be a character string. For more information about the DB2 SQL statements, refer to the IBM documents for your platform and operating system.

Examples

Insert a row into a DB2 table using DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE:

```
DECLARE
  num_rows integer;
BEGIN
  num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@dblink ('INSERT
  INTO SCOTT.DEPT VALUES (10,''PURCHASING'',''PHOENIX'')');
END;
/
```

Create a table in DB2 using DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE:

```
DECLARE
```

```
  num_rows integer;
BEGIN
  num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@dblink
  ('CREATE TABLE MYTABLE (COL1 INTEGER, COL2 INTEGER, COL3 CHAR(14),
  COL4 VARCHAR(13))');
END;
/
```

Retrieving Result Sets Through Passthrough

Oracle Transparent Gateway for DB2 provides a facility to retrieve results sets from a `SELECT` SQL statement entered through passthrough. Refer to the *Oracle Database Heterogeneous Connectivity Administrator's Guide* for additional information.

Example

```
DECLARE
  CRS binary_integer;
  RET binary_integer;
  VAL VARCHAR2(10)
BEGIN
  CRS:=DBMS_HS_PASSTHROUGH.OPEN_CURSOR@gtwlink;
  DBMS_HS_PASSTHROUGH.PARSE@gtwlink(CRS,'SELECT NAME FROM PT_TABLE');
BEGIN
  RET:=0;
  WHILE (TRUE)
  LOOP
    RET:=DBMS_HS_PASSTHROUGH.FETCH_ROW@gtwlink (CRS,FALSE);
    DBMS_HS_PASSTHROUGH.GET_VALUES@gtwlink (CRS,1,VAL);
    INSERT INTO PT_TABLE_LOCAL VALUES (VAL);
  END LOOP;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      BEGIN
        DBMS_OUTPUT.PUT_LINE('END OF FETCH');
        DBMS_HS_PASSTHROUGH.CLOSE_CURSOR@gtwlink(CRS);
      END;
    END;
END;
/
```

DB2 Data Types to Oracle Data Type Conversion

To move data between applications and the underlying database, the gateway maps data values from a host variable or literal of a specific data type to a data type understood by the underlying database.

Oracle tools and applications expect Oracle data types. Consequently, the gateway maps values from DB2 servers into appropriate Oracle data types before passing these values back to the application or Oracle tool. The data type mapping and restrictions are:

DB2 Server	Oracle External	Criteria
CHAR(N)	CHAR(N)	N=<255*
VARCHAR(N)	VARCHAR2(N) LONG	N=<4000 (refer to **Note 1**) 4000<N<DB2 maximum long value
LONG VARCHAR(N)	VARCHAR2(N) LONG	N=<4000 (refer to **Note 1**) 4000<N<DB2 maximum long value
CHAR(N) FOR BIT DATA	RAW(N)	N=<255 (refer to **Note 2**)
VARCHAR(N) FOR BIT DATA	RAW(N) LONG RAW(N)	1=<N=<2000 (refer to **Note 1**) 2000<N=<DB2 maximum long value
LONG VARCHAR(N) FOR BIT DATA	RAW(N) LONG RAW(N)	1=<N=<2000 (refer to **Note 1**) 2000<N=<DB2 maximum long value
DATE	DATE	Refer to "Performing Date and Time Operations" on page 9-12
TIME	CHAR(8)	Refer to "Performing Date and Time Operations" on page 9-12
TIMESTAMP	CHAR(26)	Refer to "Performing Date and Time Operations" on page 9-12
TIMESTAMP	TIMESTAMP(6)	Based on the ORACLE_TIMESTAMP environment variable. Refer to "Performing Date and Time Operations" on page 9-12
GRAPHIC (N)	CHAR(2N)	N<127
VARGRAPHIC (N)	VARCHAR(2N)	N<2000
LONG VARGRAPHIC (N)	VARCHAR2(2N)	2000<N<DB2 maximum long value

DB2 Server	Oracle External	Criteria
FLOAT(N) (single)	FLOAT(21)	1=<N=<21
FLOAT(N) (double)	FLOAT(53)	22=<N=<53
Decimal(P,S)	NUMBER(P,S)	n/a
INTEGER	NUMBER(10)	n/a
SMALLINT	NUMBER(5)	n/a
ROWID	RAW(40)	
BLOB	LONG RAW	
CLOB	LONG	
DBCLOB	LONG	

Note:
- In the previous table, although the limits of some data types within the Oracle database server has increased, the limits used in the gateway are not changed. This is so that you can maintain existing user application compatibility.
- Also, the Oracle database server can support a length of up to 2000 fixed character columns, but the maximum for DB2 is 255.
- The Oracle Database 10*g* database server allows only one LONG column per table. This might allow for a situation where a DB2 table cannot be directly replicated as an Oracle table.

To determine DB2 maximum long values, refer to the IBM documents for your platform and operating system.

In certain cases, the equivalent Oracle CHAR, VARCHAR or LONG columns will be different lengths than the original DB2 columns. This is due to character set differences that require more (or fewer) bytes in the Oracle representation than in the original DB2 representation.

In the most common case, the Oracle character set Unicode AL32UTF8 uses 1, 2 or 3 bytes to represent one character. If the DB2 column is single-byte EBCDIC and defined as CHAR(10), it shows up through the gateway as CHAR(30). This is required because each EBCDIC character (which only takes 1 byte in the single-byte EBCDIC character set) may require 1, 2 or 3 bytes to represent the character in Unicode AL32UTF8.

Some character sets will double the length of the original DB2 character columns, some will triple it. If the Oracle database is running in a single-byte ASCII character set, the length will always be the same in DB2 and Oracle, with one exception.

In the case of Oracle running in a single byte character set (like US7ASCII or WE8ISO8859P1) and DB2 GRAPHIC columns (which use two bytes to represent each character), the size of the Oracle character column will be half the size of the original DB2 GRAPHIC column.

Other character sets (like Unicode) will cause DB2 GRAPHIC columns to require more bytes than the original DB2 columns.

Performing Character String Operations

Frequently, DB2 databases are designed to hold non-character binary data in character columns. Applications run on DB2 systems can store and retrieve data as though it contained character data. However, when an application accessing this data runs in an environment using a different character set, inaccurate data might be returned.

When character data is sent to DB2 from an ASCII system, ASCII data is translated to EBCDIC. This translation is meaningless when the characters are binary data in a character column. The application receives incorrect information or errors.

To resolve these errors, the gateway requires character columns on DB2 holding non-character data be created with the FOR BIT DATA option. In the application, the character columns holding non-character data can be processed using the Oracle data types RAW and LONG RAW. The DESCRIBE information for a character column defined with FOR BIT DATA on the host always indicates RAW or LONG RAW.

Existing DB2 tables can be changed by directly updating the DB2 catalog.

For more information about DB2 parameters, refer to the IBM documents for your platform and operating system.

Converting Character String Data Types

The DB2 VARCHAR data type can be from one to the maximum long value for DB2. This data type is converted to an Oracle VARCHAR2 data type if it is between 1 and 4000 characters in length. If character length is between 4000 and the maximum long value for DB2, then it is converted to an Oracle LONG data type.

For additional information about determining the maximum long value for DB2, refer to the IBM documents for your platform and operating system.

The Oracle LONG data type can be from 1 to 2 G in length, but the DB2 VARCHAR data type can be no longer than 32 740 bytes. If you define a LONG data type longer than 32 740 bytes in length, then you receive an error message.

Performing Date and Time Operations

The implementation of date and time data differs significantly in DB2 databases and the Oracle database server. The Oracle database server has a single date data type, DATE, containing both calendar date and time of day information. DB2 databases support the following three distinct date and time data types:

Date or Time Date Type	Description
DATE	is the calendar date only.
TIME	is the time of day only.
TIMESTAMP	is a numerical value combining calendar date and time of day with microsecond resolution of the time value.

There is no built-in mechanism that translates the IBM TIME and TIMESTAMP data to Oracle DATE data. An application must process TIME data types in the Oracle CHAR format with a length of eight bytes. An application must process the TIMESTAMP data type in the Oracle CHAR format with a length of 26 bytes.

An application reads TIME and TIMESTAMP columns as character strings and converts or subsets portions of the string to perform numerical operations. TIME and TIMESTAMP values can be sent to a DB2 database as character literals or bind variables of the appropriate length and format.

Oracle and IBM DATE data types are mapped to each other. If an IBM DATE is queried, then it is converted to an Oracle DATE with a zero (midnight) time of day. If an Oracle DATE is processed against an IBM DATE column, then the date value is converted to the IBM DATE format and any time value is discarded.

Character representations of dates are different in the Oracle database server format and DB2 format. When an Oracle application SQL statement contains a date literal or conveys a date using a character bind variable, the gateway must convert the date to a DB2-compatible format.

DB2 Local Date Exit

Oracle Transparent Gateway for DB2 includes a DB2 local date exit. The exit is called only when needed and does not interfere with normal DB2 operations or

impact performance. With the exit installed, DB2 DATE columns are handled through the gateway. If you do not install the exit, then Oracle SQL requires changes when referencing DB2 DATE columns.

When a string constant, string bind variable, string expression, or character column is compared or assigned to a DB2 date column, it is converted from its string format to an internal DB2 format before DB2 processes it. DB2 date conversion routines look for the following formats of date string. The DB2 local date exit is called only if the date string does not match any of the standard formats.

DB2 Date Format	Pattern	Example
EUR	DD.MM.YYYY	30.10.1994
ISO	YYYY-MM-DD	1994-10-30
JIS	YYYY-MM-DD	1994-10-30
LOCAL	DD-MON-YY	30-OCT-94
LOCAL	DD-MON-RR	29-MAR-05
USA	MM/DD/YYYY	10/30/1994

The LOCAL DB2 date format is available when the gateway local date exit is installed.

The local exit is called only if the date format cannot be matched to ISO, JIS, USA, or EUR formats. In a native DB2 program, this is frequently due to a bad date string value. If a bad date string value is entered, then the DB2 local date exit is called and rejects the bad date string.

The DB2 local date exit is called in the following circumstances:

- A native DB2 program has a bad date string value that cannot be matched to ISO, JIS, USA, or EUR formats.
- A gateway program supplies one of the Oracle DATE formats, 'DD-MON-YY' or 'DD-MON-RR'.

When you install the DB2 local date exit (DSNXVDTX) supplied with the gateway, you can use ISO, JIS, USA, and EUR as well as the Oracle date formats 'DD-MON-YY', 'DD-MON-YYYY', and 'DD-MON-RR' through the gateway. The DB2 local date exit must be installed in order to specify these Oracle date formats without any SQL changes. If you do not install the exit, then you must use the Oracle TO_DATE function to pass these Oracle date formats through the

gateway. Refer to "Step 4: Make Authorization and Local Date Exits Available to DB2" on page 5-28 for further information.

Date Considerations in SQL Coding

If the gateway local date exit is installed on the DB2 system, then DB2 DATE columns appear as Oracle DATE columns through the gateway. Normal Oracle DATE processing and string values can be used. DB2 DATE columns are handled as Oracle DATE columns.

If the gateway's DB2 local date exit is not installed, then most SQL statements referencing DB2 DATE columns require changes. When a string constant or string bind variable is compared or assigned to a DB2 DATE column, a TO_DATE function must be added to the statement, enclosing the constant or bind variable.

When the DB2 local date exit is installed, the following SQL statements are accepted:

```
INSERT INTO EMP (HIREDATE) VALUES ('30-OCT-94');
SELECT * FROM EMP WHERE HIREDATE = '30-OCT-94';
UPDATE EMP SET HIREDATE = '31-OCT-94'
   WHERE HIREDATE = '30-OCT-94';
DELETE FROM EMP WHERE HIREDATE = '31-OCT-94';
```

If the DB2 local date exit is not installed, then the following SQL statements are required:

```
INSERT INTO EMP (HIREDATE) VALUES (TO_DATE('30-OCT-94'));
SELECT * FROM EMP WHERE HIREDATE = TO_DATE('30-OCT-94');
UPDATE EMP SET HIREDATE = TO_DATE('31-OCT-94')
   WHERE HIREDATE = TO_DATE('30-OCT-94');
DELETE FROM EMP WHERE HIREDATE = TO_DATE('31-OCT-94');
```

NLS_DATE_FORMAT Support

The following patterns can be used for the NLS_DATE_FORMAT:

DB2 Date Format	Pattern	Example
EUR	DD.MM.YYYY	30.10.1994
ISO	YYYY-MM-DD	1994-10-30
JIS	YYYY-MM-DD	1994-10-30
USA	MM/DD/YYYY	10/30/1994

The Oracle default format of 'DD-MON-YY' is not allowed with DB2. As a result, the gateway local date exit is provided to change the Oracle default date format of 'DD-MON-YY' or 'DD-MON-RR' to the DB2 ISO format of 'YYYY-MM-DD' before passing the date to DB2.

The following example demonstrates the most efficient way to enter and select date values in the twenty-first century:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD';
INSERT INTO EMP (HIREDATE) VALUES ('2008-07-23');
SELECT * FROM EMP WHERE HIREDATE = '2008-07-23';
UPDATE EMP SET HIREDATE = '2008-07-24'
   WHERE HIREDATE = '2008-07-23';
DELETE FROM EMP WHERE HIREDATE = '2008-07-24';
```

Oracle TO_DATE Function

The Oracle `TO_DATE` function is preprocessed in SQL `INSERT`, `UPDATE`, `DELETE`, and `SELECT WHERE` clauses. `TO_DATE` functions in `SELECT` result lists are not preprocessed.

The `TO_DATE` function is often needed to provide values to update or compare with date columns. Therefore, the gateway replaces the information included in the `TO_DATE` clause with an acceptable value before the SQL statement is sent to DB2.

Except for the `SELECT` result list, all `TO_DATE` functions are preprocessed and turned into values that are the result of the `TO_DATE` function with one exception. If `TO_DATE(column)` is coded, then it is always post-processed by Oracle because DB2 doesn't have a `TO_DATE` function like Oracle's.

All forms of the `TO_DATE` function (with one, two, or three operands) are supported.

Date Arithmetic

The following SQL expression forms do not work correctly with the gateway:

```
date + number
number + date
date - number
date1 - date2
```

The date and number addition and subtraction (`date + number`, `number + date`, `date - number`) forms are sent through to the DB2 server where they are rejected. The supported servers do not allow number addition or subtraction with

dates. Because of differing interpretations of date subtraction in the supported servers, subtracting two dates (`date1 - date2`) does not work correctly when postprocessed by the integrating server.

Note: Oracle recommends avoiding date arithmetic expressions in all gateway SQL statements.

Performing Numeric Data Type Operations

DB2 servers perform automatic conversions to the numeric data type of the destination column (such as integer, double-precision floating point, or decimal). You have no control over data type conversion, and this conversion might be independent of the data type of the destination column in the database.

For example, if `PRICE` is an integer column of the `PRODUCT` table in a DB2 database, then the update shown in the following example inaccurately sets the price of an ice cream cone to $1.00 because the DB2 server automatically converts a floating point to an integer:

```
UPDATE PRODUCT
SET PRICE = 1.50
WHERE PRODUCT_NAME = 'ICE CREAM CONE    ';
```

Because `PRICE` is an integer, the DB2 server automatically converts the decimal data value of 1.50 to 1.

Oracle ROWID Column

DB2 does not have a column equivalent to the Oracle `ROWID` column. Because the `ROWID` column is not supported, these restrictions apply:

- `UPDATE` and `DELETE` are not supported with the `WHERE CURRENT OF CURSOR` clause.
- When these statements are used in precompiler and PL/SQL programs, they rely internally on the Oracle `ROWID` function.
- Oracle fast refresh materialized views between the Oracle database server and DB2 are not supported.

  Oracle fast refresh materialized views rely internally on the Oracle `ROWID` column. However, complete refresh materialized views are supported.

Double Byte Character Set Support

Katakana is not supported in the DB2 GRAPHIC data type because the data type is double byte only. Katakana is encoded as single byte in IBM code pages 290 and 1027, and Oracle JA16DBCS and JA16EBCDIC930 character sets.

Katakana can be supported in DB2 CHAR and VARCHAR data types as mixed data if Oracle client programs and the Oracle database server are linked with NLSRTL release 2.3.4 or later. This uses the correct Katakana translation routines.

CHAR FOR BIT DATA

CHAR FOR BIT DATA is fixed length binary data in DB2. In the Oracle database server, CHAR FOR BIT DATA is converted to RAW, which is in variable length binary format.

DB2 CHAR and VARCHAR for bit data (RAW data types) are supported. Raw data in VALUES clause, WHERE clause predicate, or bind variables are treated as hexadecimal digits.

Ensure that your programs:

- Handle CHAR FOR BIT DATA as a variable length string of binary characters.
- Set the length to the maximum length of the DB2 fixed length column and pad with the binary value your program expects to be returned.

SQL Functions

One of the most important features of the Oracle Open Gateways product family is providing SQL transparency to the user and the application programmer. Foreign data store SQL functions can be categorized into three areas:

- Compatible

  SQL functions with the same meaning and results on both the Oracle database server and foreign data stores. Some examples of compatible SQL functions include:

  - AVG
  - CONCAT
  - COUNT
  - MAX

  - MIN
  - SUM
- Translated

  SQL functions that provide the same functionality but are referenced by a different name at the Oracle database server and the foreign data store. Translated SQL functions include NVL as an Oracle Database Server Function with the DB2 function of VALUE.
- Compensated

  Advanced SQL functions that are supported by the Oracle database server and that cannot be expressed or recognized by the foreign data store.

  SQL compensation in the Oracle Open Gateways enriches the semantics of the native SQL of a remote data source, such as DB2. This important feature of the gateway allows application developers and users to leverage the advanced features of the Oracle database server.

Refer to Appendix D, "Quick Reference to Oracle SQL Functions", for a listing of the Oracle database server functions. For more detailed information, refer to the *Oracle Database SQL Reference*.

Oracle Database Server SQL Construct Processing

Some gateway postprocessing considerations are explained below.

SELECT Without the FOR UPDATE Clause

A SELECT without the FOR UPDATE clause can be handled in one of three ways:

- If the entire WHERE clause of the SELECT statement is acceptable syntax for DB2, then it is given to DB2 to perform.
- If part, but not all, of the WHERE clause of the SELECT statement uses features not available in DB2, then the WHERE clause is split between the DB2 system and the Oracle database server.

  The portion of the WHERE clause acceptable for DB2 is sent to DB2. The Oracle database server post processes the results of the DB2 SELECT and applies the Oracle-specific WHERE clause elements. This results in DB2 doing as much of the WHERE clause as possible.

- If the entire WHERE clause is not acceptable for DB2, then an unqualified SELECT (without the WHERE clause) is sent for DB2, and the Oracle database server postprocesses the entire WHERE clause.

The Oracle database server postprocesses SELECT statements without the FOR UPDATE clause. Most Oracle SELECT statements are supported. One exception is the CONNECT BY clause.

SELECT FOR UPDATE, INSERT,UPDATE, and DELETE Clauses

DB2 must process the entire SELECT FOR UPDATE, INSERT, UPDATE, and DELETE clauses. The Oracle database server cannot postprocess these clauses. Only SQL that is a common subset of the Oracle database server and DB2 SQL can be used with these statements.

The following rules exist for the use of SELECT FOR UPDATE, INSERT, UPDATE, and DELETE clauses:

- Only Oracle syntax that is also valid for DB2 can be used. For DB2 SQL syntax, refer to the IBM documents for your platform and operating system.
- The following Oracle database server functions are supported with all options:
  - AVGCOUNT
  - MAX
  - MIN
  - SUM
  - TO_DATE
- The NOWAIT option of the FOR UPDATE clause of the SELECT statement is not supported.

Oracle Database Server and DB2 Differences

Please be aware of the following differences between the Oracle Database Server and DB2.

Mass Delete from a Segmented Tablespace

When using the following command from SQL*Plus:

```
DELETE FROM ABC@dblink
```

all rows are deleted from a segmented tablespace. However, DB2 occasionally sets the updated rows field to negative 1 (-1) even though more rows are actually deleted. This can cause the result from SQL*Plus to indicate an incorrect number of rows updated.

Oracle Bind Variables

Oracle bind variables become DB2 parameter markers when used with the gateway. Therefore, the bind variables are subject to the same restrictions as DB2 parameter markers. For example, the following statements are not allowed:

```
WHERE :x IS NULL
WHERE :x = :y
```

For more information about DB2 parameter marker restrictions, refer to the IBM documents for your platform and operating system.

Oracle Data Dictionary Emulation on a DB2 Server

The gateway can optionally augment the DB2 database catalogs with data dictionary views modeled after the Oracle data dictionary. These views are based on the dictionary tables in the DB2 database, presenting the catalog information in views familiar to Oracle users. The views created during the installation of the gateway automatically limit the data dictionary information presented to each user based on the privileges of that user.

Using the Gateway Data Dictionary

The gateway data dictionary views provide the gateway users with an interface (that looks like an Oracle database server interface) to the contents and use of the DB2 database. Some of these views are required by Oracle products.

You can query the gateway data dictionary views to look at the objects in the DB2 database and to determine the authorized users of the DB2 database.

All Oracle DB2 catalog views are supported in this release of the gateway. Refer to Appendix C, "Data Dictionary Views", for descriptions of Oracle DB2 catalog views.

DB2 Special Registers

You are able to access DB2 special registers using the gateway. During installation of the gateway, a DB2 view is created to access special registers. For example, to

find out the primary authorization ID being used by the gateway, run the following statement from your application:

```
SELECT CURRENT_USER FROM OTGDB2.OTGREGISTER@DB2
```

where OTGDB2 is the default qualifier of the `OTGREGISTER` view, and DB2 is the name of a database link to the gateway. Refer to Appendix C, "Data Dictionary Views", for a description of the `OTGREGISTER` view.

10

Error Messages, Diagnosis, and Reporting

This chapter discusses error messages generated by Oracle Transparent Gateway for DB2, the diagnosis of suspected Oracle database server errors, and the requirements for documenting these errors to Oracle Support Services.

For information about Oracle Database 10*g* for z/OS-specific error messages, refer to the *Oracle Database Messages Guide for IBM z/OS (OS/390)*.

This chapter includes the following sections:

Message and Error Code Processing

The gateway architecture includes a number of separate components. Any of these components can detect and report an error condition while processing a SQL statement referring to one or more DB2 database tables. An error condition can be complex, involving error codes and supporting data from multiple components. In all cases, the application ultimately receives a single Oracle database server error code on which to act.

Error conditions are represented in one of two ways:

- Mapped

  When possible, an error code from the DB2 database is converted to the Oracle database server error code associated with the same logical condition.

  Error code mapping is provided to support application designs that test for and act on specific error conditions. The set of mapped errors is limited to those associated with conditions common to most relational databases.

- Messages from the gateway

  Most gateway error conditions are reported to the application using one of the gateway error codes in the range of ORA-9100 through ORA-9199. These messages are less closely linked to specific DB2 database conditions. The message format is explained in "Interpreting Message Formats" on page 10-3.

  The ORA-9100 error code is returned for all errors for which a more specific error code does not exist. When an ORA-9100 error code is returned, the error might have been caused in the gateway by a DB2 support component on the target database system.

Mapping DB2 Error Messages to Oracle Error Messages

DB2 error messages, that is, `SQLSTATE` codes, are mapped to Oracle database server error codes. Notice that multiple DB2 `SQLCODE` can refer to the same Oracle database server error code.

Description	SQLSTATE Code	Oracle Database Server Error Code
No rows selected	02000	0
Unique index constraint violated	23505	ORA-0001
Table or view does not exist	52004 or 42704	ORA-00942
Object name greater than 18 characters and, therefore, object does not exist	54003 or 42622	ORA-00942
Insufficient privileges	42501	ORA-01031
Divisor is equal to zero	01519 or 01564	ORA-01476

Interpreting Message Formats

Error messages are generally accompanied by additional message text, beyond the text associated with the Oracle database server message number. The additional text includes details about the error.

Most gateway messages exceed the 70-character message area in the Oracle SQLCA. Use SQLGLM or OERHMS in the programmatic interfaces and the OCI that you use with the gateway to view the entire message. Refer to the *Oracle Database Messages Guide for IBM z/OS (OS/390)* for information about SQLGLM and the*Oracle Call Interface Programmer's Guide*for information about OERHMS.

Gateway messages use the following format:

```
ORA-nnnn:error message text
gateway message line(s)
ORA-2063:PRECEDING n LINES FROM dblink

where
```

Parameter	Description
nnnn	is an Oracle database server error number. If *nnnn* is between 9100 and 9199, then the message is from the gateway. If it is not in this range, then it is a mapped error message.
error message text	is the text of the message associated with the error.
gateway message lines	are additional messages generated by the gateway. The gateway messages lines are described in "Diagnosing Errors Detected by the Oracle Database Server" on page 10-4.
n	is the total number of gateway message lines.
dblink	is the name of the database link that is used to access the gateway.

Messages Generated by Oracle Transparent Gateway for DB2

The following message is generated by Oracle Transparent Gateway for DB2.

ORA-02063 preceding n lines from dblink

Examples

ORA-02063 PRECEDING 3 LINES FROM GTWLINK

Cause: Indicates an error from Oracle Transparent Gateway for DB2. The `dblink` in the ORA-2063 message indicates the name of the database link that was used to access the gateway.

Action: Refer to the text of the preceding message to determine what action is required.

ORA-28500 CONNECTION FROM ORACLE TO NON-ORACLE SYSTEM RETURNED THIS MESSAGE:
DSNT408I SQLCODE = -084, ERROR: UNACCEPTABLE SQL STATEMENT
DSNT418I SQLSTATE = 371512 SQLSTATE RETURN CODE
DSNT415I SQLERRP=DSNHAPLY SQL PROCEDURE DETECTING ERROR

Cause: Indicates an error from DB2. It is followed by messages from the DB2 database. The `n` in the ORA-02063 message indicates the total number of gateway message lines referenced in the ORA-28500 message.

Action: Refer to the text of the proceeding message to determine what action is required.

ORA-09611 DEFINITION OF TARGET SYSTEM DATA OBJECT IS INCONSISTENT

Cause: If this message is returned while attempting to run a DB2 stored procedure, then it is an indication that the definition of the DB2 stored procedure parameter list is inconsistent with the parameter list passed from the Oracle application.

Action: Correct the parameter list in the Oracle application to match the DB2 stored procedure being called.

For additional information about the DB2 messages that are included in the ORA-28500 message, refer to the IBM documents for your platform and operating system.

Diagnosing Errors Detected by the Oracle Database Server

If an error is detected by the Oracle database server, then the gateway message lines do not occur. For example, if the gateway cannot be accessed because of an Oracle Net or gateway installation problem, then the gateway message line is not present in the error message.

Another example of error messages without gateway message lines occurs when an `INSERT` statement attempts to insert data into a table, but does not include values for all the columns in the table. The following SQL statement causes an error message:

```
SQL> insert into EMP@DB2 values(9999);
ERROR at line 1:
ORA-00947: not enough values
```

The ORA-00947 message is not accompanied by gateway message lines because the error is detected by the Oracle database server. The Oracle database server obtains a description of the DB2 table before sending the `INSERT` statement to the gateway for processing. This allows the Oracle database server to detect when the `INSERT` statement is invalid.

Oracle Support Services

Oracle Support Services serves as the interface to the Oracle user community. Refer to the applicable Oracle Support Services publications for a discussion of policies and procedures for using their services.

Providing Error Documentation

During the error resolution cycle, Oracle Support Services might request that you provide them with computer-readable data. Send computer readable data, not formatted or printed data. The preferred method for providing error documentation is anonymous FTP. Please contact Oracle Support Services for instructions about how and where to provide documentation.

If you are requested to send data to the Support Center, then follow the documentation requirements provided in "General Documentation Requirements" on page 10-5. Failure to follow these requirements might result in inability to process your tape. This might delay the resolution of errors reported.

General Documentation Requirements

When you report a suspected error, you might be asked to describe the Oracle subsystem and z/OS operating system environments in detail. Provide the full version number of each component that has an error. The full version number includes important PUT levels for the z/OS system.

Before you contact Oracle Support Services, ensure that the following information is available:

- Oracle library naming conventions
- Method of accessing Oracle utilities

- Oracle subsystem name
- Full version of the Oracle gateway
- Full version of the Oracle database server client tools
- Full version of the Oracle utility
- Full version of the third party tool (if applicable)
- PUT level
- RMID of any relevant OS module

In addition to describing the Oracle operational environment, detailed documentation specific to the error might be required. This might include:

- Gateway `PARMLIB` members
- Console logs and gateway job logs
- Utility `SYSOUT`
- System diagnostic messages
- Oracle database server error messages
- System dumps
- Gateway trace data sets
- Database engine trace data sets
- Oracle Net trace data sets
- Output from the `CLIST PGMDESCC`
- Network level trace data sets for TCP/IP

Keep in mind that more than one error is often associated with a single failure. Describe all errors for the failure being reported. If your application uses Pro*C, Pro*COBOL, or another Oracle Precompiler, then ensure that your application displays or prints out all errors it encounters. Without this information, diagnosing the problem is more difficult.

Error Diagnosis

When investigating a potential Oracle gateway error, start by determining which component is failing, where it is failing, and the error category.

Components

When reporting a problem to Oracle Support Services, identify the component suspected of failure, along with its full version and correct release level.

Error Categories

Use the following error categories to describe the error:

- Documentation errors
- Incorrect output
- Oracle database server external error
- Abend
- Program loop
- Performance
- Missing functionality

Documentation Errors

When reporting documentation errors, you are asked to provide the following information:

- Document name
- Document part number
- Date of publication
- Page number

Describe the error in detail. Documentation errors can include erroneous documentation and omission of required information.

Incorrect Output

In general, an incorrect output error exists whenever an Oracle utility produces a result that differs from written Oracle documentation. When describing errors of incorrect output, you need to describe, in detail, the operation of the function in error. Be prepared to describe your understanding of the proper function, the specific Oracle documentation that describes the proper operation of the function, and a detailed description of the incorrect operation.

If you think you have found a software bug, then be prepared to answer the following questions:

- Does the problem occur in more than one Oracle tool? (Examples of Oracle tools are SQL*Plus and Oracle Developer forms.)
- What are the exact SQL statements used to reproduce the problem?
- What are the full version numbers of the Oracle database server, Oracle gateway, and related Oracle software?
- What is the problem and how is it reproduced?

Oracle Database Server External Error

Oracle database server error messages are produced whenever an Oracle gateway, server, tool, or DB2 system detects an error condition. Depending on the circumstances, error messages might be fatal or nonfatal to the utility or server.

Be prepared to identify the exact error message and message number received and the complete circumstances surrounding the error.

Abend

Any program check in an Oracle utility or the Oracle gateway address is considered an error. A full system dump is required as documentation if there is a program check.

Ensure that the system dump contains all of the private area of the Oracle gateway address space; without it, diagnosis is sometimes impossible.

System abends might or might not indicate a failure of the Oracle subsystem, depending on circumstances. The following abends are not considered Oracle database server failures:

- 013 - open failure
- 122 - canceled by operator
- 222 - canceled by operator
- 322 - CPU time exceeded
- 722 - SYSOUT lines exceeded

Program Loop

A program loop is evident when the Oracle gateway task consumes CPU time rapidly, but no actual work is performed.

Any program loop occurring within an Oracle gateway address space is considered an Oracle gateway failure. Loop conditions are rarely experienced and are considered serious errors. The initial diagnostic approach with a loop consists of a system dump. If a task is in a program loop, then ensure that the system dump includes all of the private area of the gateway address space.

Further diagnosis might be required using z/OS SLIP commands. Oracle Support Services provides specific instructions on the use of SLIP, depending on the circumstances.

Performance

Oracle system performance is determined by many factors, most of which are not within the control of Oracle. Considerations such as system load, I/O topology, network topology utilization, and DB2 resource availability and utilization, make the documentation of performance errors difficult.

Provide detailed information about the state of your environment when reporting an error. Specific documentation might include:

- CPU type and memory configuration
- Database topology
- I/O topology
- Network topology
- System workload by type
- Oracle database server workload characterization
- Query completion plans
- DB2 threads and resource information

Missing Functionality

Enhancement requests can be opened with Oracle Support Services to request the inclusion of functions and features not currently available with Oracle products. When opening an enhancement request, describe the specific feature or function to be added to the product, and provide a business case to justify the enhancement.

System Dumps

When providing documentation on suspected Oracle database server failures, it might be necessary for you to provide a system dump of the Oracle gateway or utility address spaces. Dumps are initiated through the z/OS operator interface using the DUMP and SLIP commands or automatically by the Oracle gateway if it detects a problem.

Dumps sent to Oracle Support Services as documentation for suspected errors must not be formatted. Formatted dumps cannot be used. Formatting a system dump results in a significant delay in processing reported errors, and you might have to send a new, unformatted dump.

When specifying dump parameters in response to a z/OS DUMP COMM=('') command, you must include the following specification:

```
CSA,PSA,TRT,RGN
```

System Dump Data Sets

Once a SYS1.DUMPxx data set is created, the system operator is notified whenever a dump to that data set occurs. Because all Oracle abends are dumped to SYS1.DUMP data sets and are not dynamically allocated by OSDI, you must ensure that a SYS1.DUMP data set is always available.

You must also ensure that the SYS1.DUMP data set is large enough to accommodate the gateway address space.

If a SYS1.DUMP data set is not available, then a dump might be lost.

Operator-Initiated Dumps

Operator-initiated dumps are accomplished with the z/OS DUMP command:

```
DUMP COMM=(text)
```

where text is the title you want the dump to have.

After the DUMP command has been entered, you must respond to the system WTOR with:

```
R xx,[JOBNAME=(ssn)|ASID=(nnn),]SDATA=(CSA,PSA,TRT,RGN)
```

where:

Parameter	Description
xx	is the reply identification number.
ssn	is the name of the Oracle subsystem.
nnn	is the hexadecimal address space identifier of the address space you want to dump.

GTF

You might need to use GTF as a diagnostic tool under certain circumstances. Oracle Support Services provides specific instructions if this is necessary.

11

Migration and Coexistence with Existing Gateways

This chapter is divided into two sections. The first section describes the architectural differences between MPM and OSDI with emphasis on administration and configuration considerations when migrating or upgrading to the Oracle9i version of the gateway. The second section describes how you migrate from a pre-Oracle Database 10*g* MPM gateway to the new Oracle Database 10*g* OSDI gateway.

OSDI Differences

OSDI is new Oracle software specific to IBM z/OS. It provides a z/OS execution environment for certain Oracle products. Prior to OSDI, TG4DB2 and Oracle Net were supported by separate subsystems called MPM and TNS, respectively. OSDI completely replaces both MPM and TNS with new z/OS-specific software. The supported Oracle products are the same, however, and their generic product behavior (as viewed by application programs) is unchanged.

With OSDI, z/OS subsystems are no longer associated with any single Oracle product or product instance. An OSDI subsystem can support multiple Oracle database instances, multiple gateway instances, and multiple Oracle Net services.

Summary of Changes

The following sections discuss the significant changes between an MPM and an OSDI configuration.

OSDI Subsystems

With OSDI, z/OS subsystems are no longer associated with any single Oracle product or product instance. An OSDI subsystem can support multiple Oracle database instances, multiple Oracle Net services, and multiple gateway services. An OSDI subsystem has no associated address space. Refer to "Configuring and Initializing an OSDI Subsystem" on page 11-3 for a detailed description of the differences when configuring and initializing an OSDI subsystem.

Gateway Service

An OSDI gateway instance runs under the control of the OSDI subsystem. See "Configuring a Gateway Service" on page 11-3 for a detailed description of the differences as well as special considerations when running an Oracle gateway under OSDI.

Oracle Net or Network Service

The Network Service replaces the TNS facility and also runs under the control of the OSDI subsystem. The OSDI network service simplifies the networking implementation of the Oracle database service on z/OS and is implemented to be very similar to Oracle Net on other platforms. Each gateway no longer has to maintain its own listener as a network endpoint. See the appropriate section in this chapter for a detailed description of the differences as well as special considerations when running Oracle Net under OSDI:

- Configuring Network Service on page 11-5
- Operating Network Service on page 11-6
- Computer Associates or Interlink SNS/TCPaccess Support on page 11-6
- IXCF Support on page 11-6
- Using Network Service on page 11-6

The SQL*Net V1 style cross memory connect string (for example, W:sid) has been desupported with this release of Oracle Database 10*g* for z/OS. Refer to the migration chapter of the *Oracle Database User's Guide for IBM z/OS (OS/390)* for further details.

Commands and Messages

With OSDI, both MPM and TNS have been superseded. All MPM-specific and TNS-specific commands and messages are obsolete.

Error Diagnosis and Reporting

The following items discuss the differences in the way the gateway generates trace and log output.

Trace Files Under OSDI, trace files can be written to spool files or disk data sets, as under MPM. This is specified using the TRACE_DSNAME gateway region parameter. See "TRACE_DSNAME | TDSN" on page 5-14 for a detailed description of the TRACE_DSNAME parameter.

Alert Logs Alert logs are written to a spool or disk file as specified by the `SYSPRINT` DD statement. If the `SYSPRINT` DD statement is omitted, then OSDI dynamically allocates a spool file for the alert log during service startup and writes a message to the system log identifying the system-generated DD name for the allocation. If you specify a disk data set for `SYSPRINT`, and if an error occurs while it is being written (including an out-of-space condition), then OSDI closes `SYSPRINT`, dynamically allocates a spool file, and begins writing to it.

Configuring and Initializing an OSDI Subsystem

The collection of the gateway instances, database instances, and network services managed by an OSDI subsystem is called a service group. OSDI subsystems are dynamic z/OS subsystems that do not have to be initialized at system IPL and can be initialized at any time. Once initialized, the OSDI subsystem remains in the z/OS system until the next IPL. As a dynamic subsystem, an OSDI subsystem can be activated and deactivated at will.

In contrast to MPM and TNS, OSDI-managed services cannot be executed as z/OS batch jobs or as independent started tasks, or STCs. Refer to "Postinstallation Steps" on page 4-4 for details on configuring and initializing an OSDI subsystem.

When migrating from MPM, avoid using the same subsystem name as any MPM or TNS subsystem that you have been running, even if you will not be using those subsystems after the OSDI installation. Unlike MPM and TNS, local OSDI clients do not normally need to be given, or need to specify, the OSDI subsystem name when connecting to a server. Reusing an MPM or TNS subsystem name therefore provides no particular benefit.

Configuring a Gateway Service

To run an Oracle gateway instance under OSDI, you must define the instance as a service using the OSDI `DEFINE SERVICE` command. You do this when converting an existing MPM-based instance to OSDI. In addition to defining the service, a few

other items must be set up before the service can be started: a JCL procedure, several parameter files, and possibly security resource definitions. After these are in place, you can start the service, which creates one or more address spaces based on controls that you have specified.

SID

Because OSDI gateways are not implemented as individual subsystems, OSDI gateways are no longer identified by subsystem names; instead, they are identified by SIDs. The SID is the OSDI Service Identifier that is used to identify the gateway. Under OSDI, the SID is defined during gateway service definition. A gateway service must be defined with a service name and optionally a SID. The service name is the default for the SID if the SID is not specified.

Gateway Instance JCL

A JCL procedure name is specified during gateway service definition. The JCL procedure must then be created before the gateway service is started. Note that JCL and load library changes should not be made while a gateway service is active. The PARM parameter of the EXEC JCL statement is not used by the OSDI gateway instance program.

The new ORA$FPS DD statement specifies an input file containing OSDI-specific file management parameters. Refer to "File Processing Considerations" on page 11-4 for further details.

Oracle Net Access

OSDI simplifies the networking implementation on z/OS and makes it behave in a manner similar to Oracle Net on other platforms. The Oracle Net master task MPMTNS is no longer needed in any of the OSDI gateway address spaces in order to access the OSDI gateway. For further details on configuring Oracle Net under OSDI, refer to Chapter 6, "Oracle Net". Further descriptions of Oracle Net differences under OSDI can be found in the Oracle Net or Network Service section of this chapter.

File Processing Considerations

OSDI processing of z/OS data sets that are used by the gateway differs significantly from that in MPM. The major visible changes include the following:

Allocation (creation) specifications and other file processing controls are supplied to the gateway in an input parameter file, the `ORA$FPS` DD statement. Keyword parameters are used to specify, by type of file, things such as SMS classes, allocation

unit and volume, and so on. Each of the major types of files that is used by the gateway can be separately managed: gateway trace data sets and network trace data sets.

Operating a Gateway Service

OSDI operating commands are used to start, stop, and display OSDI services. These commands can be issued through the z/OS command interface, either automatically (for example, `COMMANDxx` member of `SYS1.PARMLIB`) or manually through a console interface such as SDSF. All MPM commands and messages are obsolete and superseded by OSDI commands and messages. There is no OSDI equivalent to the `MPMCMD` command for TSO. A console interface such as SDSF must be used to issue OSDI commands from TSO. OSDI commands may also be issued through the OSDI program interface by a management component such as Oracle Enterprise manager agent. Operating commands are also permitted in the service group configuration file.

The OSDI `START` command must be used to start the gateway service. The OSDI `START` command is used to cause a gateway service to begin execution using the JCL procedure specified in the service definition. The OSDI `START` command (not the z/OS command) must be used to start services. Unlike MPM, OSDI-managed services cannot be executed as z/OS batch jobs or as independent started tasks or STCs.

Oracle Net or Network Service

The network service replaces the TNS facility and is started with an OSDI `START` command. The OSDI network service simplifies the networking implementation of the Oracle gateway service on z/OS and is implemented to be very similar to Oracle Net on other platforms.

Each gateway instance no longer has to maintain its own listener as a network endpoint. As a result, each gateway instance does not represent a distinct network address (for example, TCP/IP host name and port) that a client had to know to connect to that gateway. Clients connect to target z/OS gateway in the same manner as they connect to Oracle servers on other platforms by specifying the SID parameter that is part of the Oracle network address string.

Configuring Network Service

The Network Service is configured as an OSDI service using the OSDI DEFINE `SERVICE` command. Like other OSDI services, the Network Service is defined with a service name and SID (defaults to service name). Unlike TNS, the Network Service

SID is used to identify the Network Service to inbound clients. The Network Service SID must be four characters or less. The Oracle Net JCL procedure name must have an associated z/OS userid in order to use TCP/IP, which is controlled by z/OS UNIX System Services (USS). The associated z/OS userid must have an OMVS RACF segment (or equivalent, if a product other than RACF is used) if the installation is not using a default OMVS segment.

Operating Network Service

The OSDI `START` command must be used to start the network service. The OSDI `START` command is used to cause a network service to begin execution in its z/OS address spaces using the JCL procedure specified in the service definition. The OSDI `START` command (not the z/OS command) must be used to start services. Unlike TNS, OSDI-managed services cannot be executed as z/OS batch jobs or as independent started tasks or STCs. All TNS commands and messages are obsolete and superseded by OSDI commands and messages. Refer to Chapter 6, "Oracle Net" for details about operating Oracle Net Network Services.

Computer Associates or Interlink SNS/TCPaccess Support

SNS/TCPaccess is no longer supported as a separate Oracle Net driver. Support is provided through the HPNS interface of the IBM TCP/IP driver.

IXCF Support

IXCF is no longer supported as a separate Oracle Net for z/OS protocol. IXCF can be supported through IBM TCP/IP, in which case, Oracle Net IBM TCP/IP protocol can be used.

Using Network Service

Unlike TNS, the OSDI Network service will listen for multiple OSDI Oracle gateways on the same port number but cannot listen on multiple ports The `TNSNAMES.ORA` file entries that are being used by remote clients may need to be changed in order to reflect a new port number. Also, for OSDI, the gateway SID must be specified in the connect string for all inbound connections. See Chapter 6, "Oracle Net" for a description of configuring and administering Oracle Net with OSDI. Also refer to the migration chapter of the *Oracle Database User's Guide for IBM z/OS (OS/390)* for a detailed discussion of migration considerations for using Oracle Net with OSDI.

Migration and Upgrade

This section describes how to migrate from MPM-based gateways to the OSDI gateway. Moving to Oracle with OSDI from MPM-based Oracle and TNS-based Net is straightforward for most installations. This section of the chapter outlines the considerations in such a move, with references to detailed topic coverage in other chapters of this manual and in Oracle Database 10*g* for z/OS documentation.

Release Incompatibilities

This section lists the gateway behavior differences between releases.

Local Database Links

Local cross-memory database link access between an OSDI server and an MPM-based server or gateway is not supported. Access between local or remote OSDI and MPM servers or gateways must be performed through MPM-based SQL*Net and OSDI-based Oracle Net.

Migration and Upgrade Steps

Here are the steps required to upgrade or migrate your existing MPM-based gateway to the new Oracle Database 10*g* OSDI-based Transparent Gateway for DB2. When possible, you should follow the links back to the detailed description of each step.

Step 1: Create and Configure an OSDI Subsystem

Before any other upgrade steps will work, the subsystem must be defined to z/OS. This step is described in detail in "Postinstallation Steps" on page 4-4.

Step 2: Create an OSDI Gateway Service

Create a JCL procedure for starting the service and place it in a system procedure library PDS. (The procedure library must be one that can be accessed without a JCLLIB statement.) Make sure that an appropriate z/OS user ID will be associated with address spaces that are started by using the procedure. Create the parameter files for the gateway program and save them as members of a PDS or as DSORG=PS data sets. Requirements for the JCL and for the region parameters are described in "Step 9: Edit the PARMLIB Members" on page 5-35.

Step 3: Create and Configure OSDI Net Service

Create and configure the OSDI Net Service as described in Chapter 6, "Oracle Net". Pay particular attention to the TNSNAMES entries and ensure that entries are updated to include SID parameters.

Step 4: Establish Security

Perform installation required security steps for the OSDI subsystem and the gateway to DB2 connections. The OSDI security steps are covered in "Step 10: Associate User IDs with Services" on page 5-37. The gateway to DB2 security steps will depend on your current connection level security situation.

The 10g DB2 Gateway completely changes the way that DB2 security is done. If you modified the G4AUTH or DB2AUTH exits previously, these are no longer used.

The 10g DB2 Gateway uses the G4RRSAF Gateway logon exit, and does not modify the DB2 logon exits (as was done in the DB2 Gateway 9.2 and earlier releases).

If you had custom modifications to G4AUTH for the DB2 Gateway 9.2 or earlier and want to maintain these changes, they will have to be made to the new G4RRSAF exit. This is supplied in the SRCLIB in source form as G4SIGNON. Instructions are in chapter 7 for modifying the gateway logon exit.

Step 5: Ensure User Exits Are Available to DB2

In addition to the security exits discussed above, you must ensure that the date user exits are also available to the new gateway if they have been implemented in the current MPM-based gateway. This procedure is covered in "Step 4: Make Authorization and Local Date Exits Available to DB2" on page 5-28.

Step 6: Prepare the DB2 Environment

The DB2 environment must now be upgraded so that the Oracle packages, plan, views, and tables are at the level expected by the gateway. The necessary steps are documented in detail at:

Step 5: Run the Scripts to Create Required Tables and Views in DB2 on page 5-29

Step 6: Bind the DB2 Package on page 5-30

Step 7: Bind the DB2 Plan on page 5-32

Step 8: Grant EXECUTE on DB2 Plan on page 5-35

Step 7: Start the OSDI Services

Start the OSDI services defined in steps 2 & 3, as described in "Operation of the Gateway Subsystem with OSDI" on page 7-1.

Step 8: Test the Gateway

The last step is to create the database link as described in "Database Link Behavior" on page 8-1, and test the connection between Oracle, the gateway, and DB2.

Configuring Multiple OSDI Gateway Services

You can create multiple gateway services by defining a new service with a different service name and a different SID. The gateways will share the same subsystem, but all parameters, including the DB2 region they connect to, can be different. The steps required are a subset of the above steps.

The first step is the same as Step 2 above, paying particular attention that the service name and the SID are unique.

If your new gateway service is connecting to the same DB2 region as your existing gateway, then skip to Step 7 above. If not, then skip to Step 4 and execute each step as before.

MPM/TNS and OSDI Coexistence

You can continue to run MPM-based Oracle instances and TNS network services during your transition to OSDI. However, you must be aware of the limitation with database links (distributed access) between Oracle and/or gateway instances on the same z/OS system. Cross-memory database links (in either direction) between MPM and OSDI instances are not supported. If you have a requirement for such distributed access, then you will need to use TCP/IP instead of cross-memory access. Doing this requires running both OSDI Oracle Net and TNS with distinct TCP endpoint definitions.

TNS and OSDI NET Communication

An example of this type of MPM-to-OSDI connection is as follows:

The `TNSNAMES` entry defined to the OSDI Oracle instance to access the MPM-based gateway is:

```
G4XXMPM =
 (DESCRIPTION=
   (ADDRESS= (PROTOCOL=TCP) (HOST=MVS.TCP.IP.ADDR) (PORT=3998) (SSN=NET))
```

```
    (HS=))
```

where PORT would be the port number that MPM TNS is listening on for the MPM gateway, and SSN is the SID of the OSDI NET.

The TNSNAMES entry defined to the MPM Oracle instance to access the OSDI-based gateway is:

```
G4XXOSDI =
 (DESCRIPTION=
    (ADDRESS= (PROTOCOL=TCP) (HOST=MVS.TCP.IP.ADDR) (PORT=4998) (SSN=TNS)
    (HS=)
     (CONNECT_DATA=(SID=G4X1))
```

where PORT would be the port number that OSDI NET is listening on for all defined OSDI services in its service group, SID is the SID of the OSDI gateway, and SSN is the SSN of the MPM TNS.

The DBLINK defined to the MPM-based Oracle instance to access the OSDI-based gateway would be:

```
Create database link g4osdi connect to scott identified by
tiger using 'G4XXOSDI';
```

The DBLINK defined to the OSDI-based Oracle instance to access the MPM-based gateway would be:

```
Create database link g4mpm connect to scott identified by
tiger using 'G4XXMPM';
```

Release 10*g* Coexistence with Prior Releases of the Gateway

In release 10*g* of the Oracle Transparent Gateway for DB2, there are no DB2 security exits installed in the DB2 subsystems. Therefore, release 10g of the gateway can coexist with any prior releases. You can leave pre-10*g* DB2 security exits installed in the DB2 subsystems. They are not used by the 10*g* gateway.

The DB2 date exit (DSNXVDTX) is compatible with all prior releases of the gateway. If you want to use this optional feature, then use the one from the10g distribution in your DB2 subsystem.

12

Globalization Support

Globalization Support is a technology enabling Oracle applications to interact with users in their native language, using their conventions for displaying data.

This chapter includes the following sections:

Overview

The Oracle Globalization Support architecture is data-driven, enabling you to add support for specific languages and character encoding schemes without requiring any changes in source code.

A number of different settings in the gateway, DB2, Oracle database server, and client affect Globalization Support processing. Character settings of these components must be compatible so that translations occur correctly. Each character in one encoding scheme must have a matching character in the other encoding schemes.

Obsolete NLS Parameters

These NLS parameters were used in earlier releases of the DB2 Gateway (before 10.2) but are obsolete now for the DB2 Gateway only. Other gateways may still continue to use these parameters:

- NLS_LANG
- NLS_NCHAR
- NLS_DATE_FORMAT
- HS_LANGUAGE
- HS_NLS_NCHAR
- HS_NLS_DATE_FORMAT

Do not specify any value for these parameters. This will cause failures in the DB2 Gateway.

If you are converting from a earlier release of the DB2 Gateway, do not use the parameters from that release. Use the parameters that are generated by the installation process as there are new required parameters that will cause the gateway to fail if they are missing, and obsolete parameters that must no longer be specified.

DB2 Character Sets Handled Automatically

Internally, the DB2 Gateway handles all DB2 Coded Character Set IDs (CCSIDs) automatically. The gateway detects the CCSIDs of DB2 columns and stored procedure parameters and handles all conversions with no parameter specification required by the user.

First, a little background on character set handling in the DB2 Gateway.

Unicode is the name for a family of industry-standard character sets. The Unicode character sets are designed to be all encompassing, that is, they can represent every code point in every character set. This makes Unicode an ideal lingua-franca (common language (or common character set in this case)) for exchanging electronic data.

The interesting Unicode character sets (there are many) are normally referred to as UTF8 (AL32UTF8 is Oracle's Globalization Support character set for Unicode Version 4 UTF8) which is a variable width character set (each character set is represented by 1 to 3 bytes), and UCS2 (AL16UTF16 is Oracle's Globalization

Support character set for Unicode Version 4 UCS2) which is a fixed-width 16-bit (double-byte) character set.

With DB2 7.1 and later, now supporting EBCDIC, ASCII, and UNICODE tables, plus single-byte, mixed and double-byte CCSIDs for EBCDIC, ASCII, and UNICODE (9 CCSIDs for each DB2 subsystem), a different approach had to be taken to fully support all character sets.

The way the DB2 CCSIDs are handled is different depending on what is being done.

All SQL statements going into the gateway are converted to Unicode (AL32UTF8) before being sent to DB2. This includes passthrough SQL statements. If the Oracle server is already running AL32UTF8, no conversion is needed. So, it is highly recommended to run the Oracle server in AL32UTF8.

All input bind variables for SQL (for SELECT, INSERT, UPDATE and DELETE) are converted to Unicode (AL32UTF8). All bind variables for passthrough SQL are also converted to Unicode (AL32UTF8).

This approach of converting all SQL statements and bind variables to Unicode (AL32UTF8) allows all possible code points to be sent to DB2. No matter what character sets DB2 is running in, all code points for DB2's character sets can appear in constants in SQL statements and bind variables for SQL.

SELECT results are handled differently. The results of SELECT or passthrough SELECT statements are fetched from DB2 in the character set of the column with no character set translation. The data is then translated directly to the character set of the Oracle server. This minimizes the number of translations, and in the case of selecting DB2 Unicode data with an Oracle server running in Unicode, no character set translations take place. The same would be true for selecting DB2 ASCII data with an Oracle server running in the same ASCII character set.

There is one exception to how SELECT results are handled, and that is for the DB2 GRAPHIC data types (GRAPHIC, VARGRAPHIC and DBCLOB). These data types are always fetched in UCS2 (AL16UTF16) for easier handling internally by the gateway. The GRAPHIC column data is then translated to the character set of the server.

Stored procedures and user-defined functions (UDFs), because they require parameters of a specific length for both input and output, require a different approach.

Input, output and in/out parameters to stored procedures and UDFs are translated from the character set of the Oracle server to the CCSID of that DB2 parameter on input, and from the CCSID of the parameter to the character set of the Oracle server on output. If a stored procedure is defined to use Unicode parameters and the

Oracle server is running in Unicode, then no translation takes place. The same would be true for ASCII parameters in the same character set as the Oracle server.

Because the gateway can now handle DB2 data in any character set, the character set of the Oracle server now becomes the limiting factor for character set support. If your Oracle server was installed with the default database character set of US7ASCII, then only the 26 Latin alphabetic characters are supported. If you use a regional character set (for example- Western Europe – WE8ISO8859P1), then you will get support for all the characters in the 10 Western European languages that this character set supports. Oracle recommends the use of an Oracle server with a database character set of AL32UTF8, because this character set supports every other character set in existence. If your Oracle server is using UTF8, then it should be change to use AL32UTF8; otherwise character set translations will occur between the Oracle server and the DB2 Gateway.

If you are using Oracle Database 10g for z/OS as the integrating server, then refer to the next section, "Oracle Database 10g for z/OS," for further details, prior to installing the gateway and Oracle Database 10g for z/OS.

Oracle Database 10*g* for z/OS

Beginning with Oracle Enterprise Edition for OS/390 release 7.3.2, the Oracle database server is created with the default character set WE8EBCDIC1047 to facilitate compatibility with the IBM z/OS compiler character set.

When Oracle Enterprise Edition for OS/390 release 7.3.2 or later is used as the integrating server in your Oracle Transparent Gateway for DB2 configuration, it is important (when creating the Oracle Database 10g for z/OS database) to consider using the same EBCDIC character set for Oracle Database 10g on z/OS as the DB2 EBCDIC installation CCSID. In most cases, the DB2 CCSID is 37 (WE8EBCDIC37 in Oracle Globalization Support character sets). It is important to confirm this setting because the default setting for the Oracle database character set on z/OS is WE8EBCDIC1047, which is IBM's EBCDIC 1047 that is not a supported CCSID for DB2.

If the character sets vary between the Oracle Database 10g for z/OS and the DB2 EBCDIC installation CCSID, then performance can be impacted because all data requires translation. The Oracle Database 10g for z/OS must run in some EBCDIC database character set, Unicode (AL32UTF8) cannot be used for the database character set on z/OS. In addition, the NLS_LANG specification of a z/OS client process (such as SQL*Plus or client application programs) should be the same for the Oracle Database 10g for z/OS database to avoid translation between the client process and Oracle Database 10g for z/OS.

Double-Byte Character Support

All DB2 GRAPHIC (double-byte) data types (GRAPHIC, VARGRAPHIC, and DBCLOB) are automatically mapped to their Oracle CHAR equivalents. Mapping of DB2 GRAPHIC data types to Oracle NCHAR data types is no longer supported. There is an environment variable that controls certain aspects of the double-byte character set support, specified in the ORA$ENV DD.

The following describes the parameter 'ORADBMBOPT'.

This version of the Oracle Transparent Gateway for DB2 allows you to access and change columns in DB2 that are designated as VARGRAPHIC or GRAPHIC. Because of the IBM definition of a (VAR)GRAPHIC column, you can insert only double-byte characters into (VAR)GRAPHIC columns.

Some of IBM's double-byte character sets (the EBCDIC ones) do not have a double-byte character that represents the regular letters A-Z. They only have "double-wide" versions of A-Z which are a set of different characters.

All of IBM's ASCII double-byte character sets and IBM's Unicode double-byte character set do not have this problem. They have a representation for the regular letters A-Z and for a double-wide A-Z.

Because the DB2 Gateway provides all SQL statements and SQL bind variables to DB2 in Unicode (AL32UTF8), any single or double-byte character can be sent to DB2 in literals or bind variables, and DB2 automatically converts single-byte characters to their double-byte equivalent in those character sets that do not have a representation for single-byte characters.

Inserting a value represented by a mixed-byte string constant into a (VAR)GRAPHIC column defined to use an EBCDIC character set will result in automatic conversion by DB2 of single-byte only characters to their double-byte equivalents.

For example, an ASCII client issuing the following `INSERT` statement:

```
INSERT INTO mytable.graphcol@tg4db2 values('AxxB')
```

Where `A` and `B` are the normal ASCII single-width characters and `xx` designates a double-width character in one of the Asian ASCII-based character sets such as JA16SJIS, ZHS16CGB213280, ZHT16BIG5, or KO16KSC5601. Where no double byte character corresponds to a single-byte character, it is converted by DB2 to its double-byte equivalent. In the above example, `A` and `B` would be stored as double-wide `A` and `B`.

When selecting back the above values, it is unclear when we find a double-wide letter whether it should be converted back to a regular letter, or whether it should be returned as a double-wide letter (for example, Unicode AL32UTF8 has encodings for both a regular A and a double-wide A).

This version of the Oracle Transparent Gateway for DB2 provides a workaround, but the workaround carries a risk for data integrity. If you desire to choose this path, then you must realize that if you attempt to INSERT a string with a single-byte character, then that string may be returned to you entirely as a DBCS string in a subsequent SELECT operation as a double-wide character. The rules are as follows:

- On input, regardless of ORADBMBOPT option, the single-byte characters would still be converted to their double-byte equivalents before being inserted to DB2 graphic datatypes. This is different from 10gR1 or prior gateway in which the single-byte characters would be inserted to DB2 graphic datatypes as is, if ORADMBOPT is not set to FORCE_SB. The same is true for values represented by constants or for values represented by bind variables.
- On output, ORADBMBOPT option would make a difference. If ORADBMBOPT=FORCE_SB is specified, then on a SELECT, each (VAR)GRAPHIC column will be examined for double-byte characters that correspond to single-byte characters. Each such character is replaced by its corresponding single-byte character. This is where the data integrity problem arises. Because single-byte characters have been forced to their corresponding double-byte characters on input, there is no way to know if a double-byte character actually came from the translation of an ASCII wide character or from the process of forcing single-byte characters to the corresponding double-byte characters. If ORADBMBOPT is not set to FORCE_SB, then the double-byte characters would not be replaced by its corresponding single-byte characters.

As an example, use the previous INSERT statement, which is:

```
INSERT INTO mytable.graphcol@tg4db24 values('AxxB')
```

The resultant DBCS value in the DB2 column may look like the following:

```
wAyywB
```

where wA is the DBCS correspondence (usually 0x42C1) for the double wide A, wB is the DBCS correspondence (usually 0x42C2) for the double wide B, and yy is the DBCS character corresponding to the ASCII-based xx. When SELECTing from this column, on the client you would get exactly what you INSERTed, that is, AxxB—which is good. But what if the DBCS character that is represented in the DB2 column by wa had actually been INSERTed into the column through a valid ASCII

representation for a wide A? With the FORCE_SB option, you get a single-byte A on the client. This may not be exactly what you wanted.

In the end, it is you who must decide whether this option is valuable to you. You must decide if you can accept the possible problems that can arise.

Some character sets will get no double-wide characters returned (like US7ASCII). All double-wide characters will be returned as the substitution character for that character set (usually a?). Unicode (AL32UTF8) can represent both regular and double wide letters so if you run the Oracle server with that character set, then you may not want to use this option.

The ORADBMBOPT environment variable controls this feature. If there is no ORADBMBOPT environment variable, then no forcing of double-byte characters to the single-byte correspondences will take place when transferring data from DB2 columns. Placing the string FORCE_SBCS into ORADBMBOPT turns the feature ON.

One good workaround for this double-byte problem is to use Unicode for all your DB2 GRAPHIC columns. The double-byte Unicode includes code points for both regular and double-wide characters, so anything can be inserted into these columns and it will come back out the same.

Note that you should not use passthrough to execute any SQL commands that contain graphic constants, unless those constants conform fully to DB2 graphic constants. DB2 graphic constants start with G' (G apostrophe) or N' (N apostrophe) and end with an apostrophe (') character. These will be converted to Unicode (AL32UTF8), and the constants must follow the DB2 rules for forming Unicode literals.

Oracle Database Server and Client Configuration

There are a number of NLS parameters controlling Globalization Support processing between the Oracle database server and client. You can set language-dependent behavior defaults for the server and set language-dependent behavior for the client that overrides these defaults. For a complete description of NLS parameters, refer to the Globalization Support chapter in the Oracle Database SQL Reference. These parameters do not affect the gateway processing. However, you must ensure that the character set is compatible with the DB2 data's CCSIDs. In other words, each character in one encoding scheme must have a matching character in another encoding scheme.

When you create the database, the character set used to store data is specified by the CHARACTER SET parameter. Once the database is created, the database character set cannot be changed unless you re-create the database.

Normally, the default for CHARACTER SET on non-EBCDIC platforms is US7ASCII, which supports only the 26 Latin alphabetic characters. If you have specified 8-bit character sets on the gateway and DB2, then you must have a compatible 8-bit character set defined for your database. To check the character set of an existing database, enter the command:

```
SELECT USERENV('language') FROM DUAL;
```

Message Availability

The gateway itself never returns any message text, so the setting in NLS_LANG in the DB2 Gateway for language is not used. In fact, NLS_LANG should never be specified for the DB2 Gateway.

All message text is retrieved by the Oracle server, so set your language parameters on the server or the client side to see the message in a different language.

In some cases, DB2 error message text is embedded within an Oracle error message. DB2 supplies error message text only in English, so the DB2 error message text is never translated to other languages.

A

OSDI Subsystem Command Reference

OSDI provides a set of system commands for defining and controlling instances of Oracle products. This appendix is the primary reference for all OSDI commands.

This appendix includes the following sections:

OSDI Command Reference

OSDI provides a set of system commands for defining and controlling instances of Oracle products. This appendix is the primary reference for all OSDI commands.

Command Types and Processing

OSDI commands are broadly divided into definition commands and operating commands, as follows:

- Definition commands are used to create and manipulate data structures that describe service groups and services. These commands commonly appear in the subsystem configuration file.
- Operating commands are used to manage execution of services.

Three mechanisms exist for issuing OSDI commands:

- Subsystem configuration file, processed during subsystem initialization
- System operator command interfaces for z/OS (system consoles, SDSF, SYS1.PARMLIB (COMMND*xx*), Netview, and others)
- OSDI program interface, used internally by Oracle products.

When an OSDI command is issued using a z/OS system operator command interface, the target subsystem is specified by using the command prefix associated with the subsystem. When the program interface is used to issue an OSDI command, the target is identified by its subsystem name rather than a command prefix. Commands in the configuration file always apply to the subsystem (service group) being initialized, and they must omit the prefix.

Commands generally result in synchronous response messages ranging from simple acknowledgment to multiline displays. Responses to commands that are issued through system operator facilities normally are directed to the issuing console. Various operating commands can result in subsequent, asynchronous messages. These messages are not necessarily directed to the console or session that issued the original command.

System Symbols in Commands

To meet the requirement that the particulars of a service (that runs on multiple systems in a sysplex) can be tailored by system, z/OS system symbols (alphanumeric names prefixed with the ampersand character, "**&**") can be used in the specification of certain OSDI command parameters. These command parameters resolve to system-specific or IPL-specific values set by z/OS or by the installation. If a command parameter can include system symbols, this capability is noted in the parameter description.

Definition Commands

Definition commands are used to create, modify, and display the data structures of the service group. An initial set of commands in the configuration file directs the building of these structures during subsystem initialization. Subsequent definition commands can be used to add new service definitions, modify existing definitions, and so on. The overall data structure persists for the life of the IPL.

The definition commands operate only on data structures; they do not directly affect the operation of services.

Three definition commands are supported:

- `DEFINE` - Create a logical structure.
- `ALTER` - Modify the definition of an existing logical structure.
- `SHOW` - Display contents of an existing logical structure.

> **Note:** `SHOW` deals with definition data only and is distinct from the operating command, `DISPLAY`. Refer to "SHOW" on page A-5, "SHOW" on page A-10, and "DISPLAY" on page A-12.

Structures

The definition commands operate about the following structures:

- `SERVICEGROUP` - The primary structure of the subsystem.
- `SERVICE` - A structure representing an instance of an Oracle product.

The various parts of these structures are generally referred to as attributes. Definition commands use keywords to identify attributes being set or modified.

Service Group Definition Commands

The following sections provide information about Service Group Definition Commands.

DEFINE

`DEFINE SERVICEGROUP` must be the first command issued to a newly initialized subsystem; normally, it appears in the configuration file just after the bootstrap

(INIT) record. DEFINE SERVICEGROUP must be processed successfully in order for any other OSDI commands or functions to be usable.

The command structure for defining a service group is shown in the following example:

```
DEFINE SERVICEGROUP
   [ SSID( string ) ]
   [ DESCRIPTION( string ) ]
   [ MODE( LOCAL | SYSPLEX | *SYS ) ]
   [ SYSTEMS( sysname [ sysname... ] | *ALL ) ]
```

Define Parameters

Parameter	Description
SSID	Specifies the one-character to four-character z/OS subsystem name associated with the service group. If coded, then it must match the subsystem identifier specified in the IEFSSN*xx* parmlib member or the SETSSI operator command. This parameter defaults to the correct value. It is therefore coded only to confirm that the correct configuration file is in use. *String* cannot contain system symbols.
DESCRIPTION	Specifies an arbitrary text string of up to 64 characters that appears in certain displays associated with the service group. This can be used to supply installation-specific identification for the service group. The default value is 'Service Group ssn' where *ssn* is the subsystem name. *String* can contain system symbols but should not contain non-printable characters or control characters.
MODE	This parameter is not yet supported.
SYSTEMS	This parameter is not yet supported.

ALTER

ALTER SERVICEGROUP is used to modify attributes of a service group. It can be included in the configuration file, and it can be issued after initialization. Not all attributes can be altered. The subsystem ID, for example, is constant for the life of the IPL.

The command structure for altering a service group is shown in the following example:

```
ALTER SERVICEGROUP
```

```
[ DESCRIPTION( string ) ]
[ MODE( LOCAL | SYSPLEX ) ]
[ SYSTEMS( sysname [ sysname... ] | *ALL ) ]
```

Alter Parameters

Parameter	Description
DESCRIPTION	Specifies an arbitrary text string of up to 64 characters that appears in certain displays associated with the service group. This can be used to supply installation-specific identification for the service group. *String* can contain system symbols but should not contain non-printable or control characters.
MODE	This parameter is not yet supported.
SYSTEMS	This parameter is not yet supported.

SHOW

SHOW SERVICEGROUP is used to display the current definition of the service group. It can be included in the configuration file, and it can be issued after initialization. The command for displaying a service group definition is shown in the following example:

```
SHOW SERVICEGROUP
   [ LONG ]
```

Show Parameters

The show parameter is LONG, which specifies that the name, type, and description of each service in the service group be included in the display.

Service Definition Commands

The following sections provide information about service definition commands.

DEFINE

DEFINE SERVICE is used to create a structure for executing an installed Oracle product. It can be included in the configuration file, and it can be issued after

initialization. Once a service is defined, it can be started, stopped, etc. using operating commands.

The command structure for defining a service is shown in the following example:

```
DEFINE SERVICE
     name
     TYPE( string )
     PROC( string )
   [ DESCRIPTION( string ) ]
   [ PARM( string ) ]
   [ MAXAS( number ) ]
   [ SID( string ) ]
   [ JOBNAME( string ) ]
   [ JOBACCT( string ) ]
   [ MODE( SYSPLEX | LOCAL ) ]
   [ SYSTEMS( sysname [ sysname... ] | *SVG ) ]
```

Define Parameters

Parameter	Description
name	Specifies the name of the service. The name is used in other commands that operate on the service or its definition and by program functions that interact with the service. It must be one to eight characters long. (Although OSDI permits service names up to eight characters long, the name you use for a database service should be seven characters or less due to a length limitation on what is stored in the database control file.) In addition, it must consist of upper case alphabetic, numeric, and/or national characters, and must begin with an alphabetic character. It must be unique within the service group. *Name* cannot contain system symbols. **Note:** Unless you specify JOBNAME, *name* is used as the job identifier when the service is started. In this case, *name* must not be the same as any subsystem name in your system.
TYPE	Specifies the Oracle product being configured. *String* is a one-character to four-character alphabetic and/or numeric identifier that designates an installed Oracle for z/OS product managed under this architecture. Current supported values are ORA (Oracle database) and NET (Oracle Net). *String* cannot contain system symbols.

Parameter	Description
PROC	Specifies the member name in a system JCL procedure library used to start an address space for the service. Usually, this is a procedure that is created during installation of the associated Oracle for z/OS product. *String* cannot contain system symbols.
DESCRIPTION	Specifies an arbitrary text string of up to 64 characters that appears in certain displays associated with the service. This can be used to supply installation-specific identification for the service. The default value is '*Service svc Type type*', where *svc* is the service name and *type* is the type. *String* can contain system symbols but should not contain non-printable or control characters.
PARM	Specifies a parameter string passed to the service when an address space is started. *String* can be from 0 characters to 64 characters. If it contains characters other than alphabetic, numeric, and national characters, then it must be enclosed in single apostrophes. To indicate a zero-length (empty) parameter, specify PARM(''). Content requirements for this string depend on the service type and are discussed in Chapter 5, "Configuring a Gateway Service" and Chapter 6, "Oracle Net". The default is a null string. The value specified can contain system symbols.
MAXAS	Specifies the maximum number of address spaces that can be started for the service. This is meaningful only for a database (TYPE(ORA)) service. If specified for any other type, then *number* must be 1. For a database service, *number* must be between 1 and 255 inclusive. The default for this parameter is 1. The value cannot include system symbols.
SID	Specifies a unique identifier for the service that is used in client- or application-supplied service addressing. *String* is a one-character to eight-character identifier that conforms to the same rules as those for service names. The value supplied must be unique within the z/OS image: it cannot duplicate the SID of any other service in this or any other service group. This parameter defaults to the service name. If you accept the default, then it means that the service name must be unique within the z/OS image. **Note:** If you migrate an MPM-based database to OSDI and are supporting pre-OSDI local client applications, then you probably want SID to match the subsystem name you were using with MPM.

Parameter	Description
JOBNAME	Specifies a z/OS jobname to use when starting address spaces for the service. *String* is either a one-character to eight-character identifier that conforms to z/OS jobname requirements, or it is a one-character to five-character identifier followed by an asterisk. The asterisk form can be used with multiple address space services to provide unique jobnames; it is replaced with a three-digit address space counter (001, 002, and so on) as each address space is started. The jobname should conform to any rules or requirements specific to your installation. This parameter has no default. If you omit it, service address spaces are started without a jobname but with the service name as the address space identifier. System symbols cannot be used in this parameter.
JOBACCT	Specifies an installation-specific string containing job accounting fields. *String* can be from 1 character to 64 characters. If it contains characters other than alphabetic, numeric, and national characters, then it must be enclosed in single apostrophes. Use this parameter if your installation requires job accounting data to be included with started tasks (STCs). If this parameter is omitted or specified as a null string (a pair of adjacent apostrophes), then no job accounting data is supplied when service address spaces are started. *String* can include system symbols.
MODE	This parameter is not yet supported.
SYSTEMS	This parameter is not yet supported.

ALTER

ALTER SERVICE is used to modify attributes of a defined service. It can be included in the configuration file, and it can be issued after initialization. The name, type, maximum address spaces, and SID of a service cannot be altered.

OSDI does not prohibit altering the definition of a running service. This enables some useful capabilities, but it may also be harmful if misused. For example, changing the JCL procedure of a running multiple address space service would have unpredictable consequences when additional auxiliary address spaces are started.

The command structure for altering a service is shown in the following example:

```
ALTER SERVICE
      name
    [ DESCRIPTION( string ) ]
```

```
[ PROC( string ) ]
[ PARM( string ) ]
[ MAXAS( number ) ]
[ JOBNAME( string ) ]
[ JOBACCT( string ) ]
[ MODE( SYSPLEX | LOCAL ) ]
[ SYSTEMS( sysname [ sysname... ] | *SVG ) ]
```

Alter Parameters

Parameter	Description
name	Specifies the name of the service to be altered. It must be the name of an existing service in the service group. *Name* cannot contain system symbols. **Note:** The name of a service cannot be altered.
DESCRIPTION	Specifies an arbitrary text string of up to 64 characters that appears in certain displays associated with the service. This can be used to supply installation-specific identification for the service. *String* can contain system symbols but should not contain non-printable or control characters.
PROC	Specifies the member name in a system procedure library used to start an address space for the service. Usually, this is a procedure that is created during installation of the associated Oracle for z/OS product. *String* cannot contain system symbols. **Note:** Altering PROC while a multiple address space or multiple system service is running can have unpredictable effects.
PARM	Specifies a parameter string passed to the service when an address space is started. Requirements for this string depend on the service type and are discussed in the associated Oracle for z/OS product documentation. The value specified can contain system symbols. **Note:** Altering PARM while a multiple address space or multiple system service is running can have unpredictable effects.

Parameter	Description
MAXAS	Specifies the maximum number of address spaces that can be started for the service. This is meaningful only for a database (TYPE(ORA)) service. MAXAS is accepted on an ALTER SERVICE command only when the service is not active. If the service is active, starting, or stopping, then an ALTER SERVICE command with MAXAS specified will be rejected.
JOBNAME	Specifies a jobname to use when starting service address spaces, as discussed under DEFINE SERVICE. You can nullify (remove) a service jobname specification by coding JOBNAME(''). The value specified cannot contain system symbols.
JOBACCT	Specifies job accounting data to be included when service address spaces are started, as discussed under DEFINE SERVICE. You can nullify (remove) job accounting data by coding JOBACCT(''). The value specified can contain system symbols.
MODE	This parameter is not yet supported.
SYSTEMS	This parameter is not yet supported.

SHOW

The SHOW SERVICE command is used to display the current definition of a service.

The command for displaying a service definition is shown in the following example:

```
SHOW SERVICE
     name
```

Show Parameters

The show parameter is name, which specifies the name of the service whose definition is to be displayed. It must be the name of an existing service in the service group. *Name* cannot contain system symbols.

Operating Commands

Operating commands manage the execution of services. They are normally issued through the z/OS command interface, either automatically (for example, COMMND*xx* member of SYS1.PARMLIB) or by a real operator. They might also be issued through the OSDI program interface by a management component such as Oracle

Enterprise Manager Agent. Operating commands are also permitted in the service group configuration file.

Available Commands

Five operating commands are provided:

- `START` - Start execution of a service.
- `DISPLAY` - Display operating status of a service.
- `DRAIN` - Place a service in quiescent state.
- `RESUME` - Restore a service to normal operation.
- `STOP` - Stop execution of a service.

All the operating commands take a service name as the first positional parameter. This service name must be the name of a defined service.

Commands

The following sections provide information about commands.

START

The `START` command initiates execution of an address space for a specified service. For a database service, this can be the first address space (service not previously started) or an auxiliary address space (service previously started and initialized successfully but not yet at its maximum address spaces). For other types of services the service must not already be running.

The command structure for starting a service is shown in the following example:

```
START
    name
   [ PARM( string ) ]
```

START Parameters

Parameter	Description
`name`	*`Name`* specifies the name of the service to start. It must be a defined service whose current state is **inactive** or, if **active**, must not already have its maximum address spaces running.

Parameter	Description
PARM	Specifies a parameter string passed to the service when an address space is started. This overrides any PARM value established by DEFINE or ALTER SERVICE. Requirements for this string depend on the service type and are discussed in Chapter 5, "Configuring a Gateway Service" and Chapter 6, "Oracle Net". *String* can contain system symbols. **Note:** A PARM override must not be used when starting auxiliary address spaces for a database service.

DISPLAY

The DISPLAY command displays execution status information for services. OSDI displays the current operating state of the service. If the service state is "active" or "drained", then the command is also posted to the running service for further processing at the service's discretion.

The command structure for displaying a service is shown in the following example:

```
DISPLAY
     name
   [ LONG ]
```

DISPLAY Parameters

Parameter	Description
name	Specifies the name of the service to be displayed.
LONG	Specifies that a more detailed display is desired. This provides information about each active address space of the service.

DRAIN

The DRAIN command places a running service in a quiescent state in which it no longer accepts new connection (bind) requests. Existing connections or sessions are not affected. The command is also posted to the running service for further discretionary processing. This command has no effect when the service is not running.

The command structure for draining a service is shown in the following example:

```
DRAIN
     name
```

DRAIN Parameters

The DRAIN parameter is name, which specifies the name of the service to be made quiescent. This must be the name of a running service whose current state is **active**.

RESUME

The RESUME command reverses the effect of a drain, allowing a service to begin accepting new connection requests. The command is also posted to the running service for further discretionary processing. This command has no effect when the service is not running.

The command structure for resuming a service is shown in the following example:

```
RESUME
     name
```

RESUME Parameters

The RESUME parameter is name, which specifies the name of the service to be resumed. This must be the name of a running service whose current state is **drained**.

STOP

The STOP command requests termination of a running service. The normal mode of stop changes the service state to **stopping** and posts the stop request to the service; it is up to the service to comply, presumably after allowing current requests to complete and performing required cleanup. A **force** option causes the service address spaces to be terminated involuntarily. The **force** form of stop requires that a normal stop be issued first. This command has no effect when the service is not running.

The command structure for stopping a service is shown in the following example:

```
STOP
     name
   [ FORCE ]
```

STOP Parameters

Parameter	Description
name	Specifies the name of the service to be stopped. This must be the name of a running service whose current state is **active**, **drained**, or (if the FORCE option is specified) **stopping**.
FORCE	Specifies that an involuntary stop is requested.

OSDI Command Keyword Abbreviations

The abbreviated or alternative forms that can be used for OSDI command verbs and parameter keywords are as follows:

```
ALTER             ALT
DEFINE            DEF
DESCRIPTION       DESC
DISPLAY           DIS, D
DRAIN             DR
FORCE             (none)
JOBACCT           ACCT
JOBNAME           JOB
LONG              L
MAXAS             MXA
MODE              MD
PARM              P
PROCEDURE         PROC
RESUME            RES
SERVICE           SRV, SVC
SERVICEGROUP      SVG, SG
SHOW              SH
SID               IDENTIFIER, ID
SSID              (none)
START             ST, S
STOP              P
SYSTEMS           SYS
TYPE              TY
```

B

The Oracle SMF Interface

The IBM System Management Facility (SMF) provides a facility for users to collect and record a variety of system and job-related information. SMF formats the information into a number of records. By creating analysis and report routines, installations can use the information in SMF records to track system usage.

The gateway service uses the standard SMF interface to write user records to SMF data sets. These user records contain gateway service accounting and information allowing gateway installation sites to charge individual users for the resources they use.

DB2 also uses SMF to collect and record data. DB2 uses record types 100, 101, and 102. Refer to your DB2 documentation for more information.

This appendix includes the following sections:

Activating SMF Records

SMF recording is activated by updating the `SMFPRMxx` member of `SYS1.PARMLIB` to include the `SYS` or `SUBSYS` option. These options record the gateway user record type. If the `SUBSYS` option is used, then the `SUBSYS` name must match the OSDI subsystem name specified by the OSDI startup parameter. Refer to the IBM documentation on System Management Facilities for information about implementing SMF.

Specifying the Oracle Gateway Record Type

The default gateway user record type is 0 (zero). A zero for this parameter indicates that no SMF statistics record is to be written. You can override the default to any value between 128 and 255 by adding the SMF_STAT_RECNO (abbreviation is SMFSTRCN) to the OSDI gateway region parameter file. The SMF record number that is chosen must not be the same as the number that is used by any other z/OS software.

Oracle recommends using SFM record number 199, but any available record number between 128 and 255 may be used. Due to the differences in the report formatting programs, OSDI gateways and the MPM gateway should use different SMF record numbers.

If this parameter is not specified, or if zero is specified, then no SMF statistics connection or recording is done. This saves some CPU overhead and saves the overhead of the SMF write itself (which is mostly asynchronous work done by the SMF address space, and the in-line overhead is mostly just moving data into SMF buffers).

Using the OSDI SMF_STAT_RECNO Parameter

To override the default record type in the SMFPRMxx member, use the following syntax:

```
SMF_STAT_RECNO | SMFSTRCN
```

SMF_STAT_RECNO can be added as an OSDI parameter to the OSDI gateway region parameter data set to override the default record number 0. In the following example, 199 is the new SMF record type:

```
SMF_STAT_RECNO(199)
```

Starting SMF Recording

SMF recording of gateway accounting information starts automatically at startup if SMF is activated and the gateway record type is specified in the SMFPRMxx member.

Because the standard system default record types activated for SMF are 128 through 255, and because the recommended value for the gateway service (199) is within this range, many sites automatically begin SMF recording of gateway records when the gateway is installed.

If the SMF_STAT_RECNO is added or modified in the OSDI gateway region parameter file while the gateway is active, then the service must be stopped and restarted for the parameter to take effect.

Stopping SMF Recording

The OSDI SMF_STAT_RECNO parameter can be used to stop SMF recording for Oracle. To stop SMF recording for Oracle regardless of what your system tables specify, use:

```
SMF_STAT_RECNO (0)
```

or take the default of 0 (zero). The service must be stopped and restarted for this parameter to take effect.

Events Generating SMF Records

After SMF recording is turned on, an SMF record is written each time a user logs off (normal termination or SMFINV set to SMFNORM), provided SMF is activated when the user logs on.

SMF records are also written on an abend or cancellation of a job if SMFINV=SMFABORT.

If the z/OS system crashes, then SMF records are not written and the information is lost.

Interpreting an Oracle SMF Record

To interpret an Oracle SMF record, you first need to dump the SMF data set to a sequential data set. You can then write a program that does all of the following:

- Reads the sequential data set
- Selects only records with the Oracle record number
- Accesses the Oracle SMF record fields using the provided DSECT
- Prints the statistics

A sample program named ORAFMTO is provided in the SRCLIB library that you can customize for your installation. Refer to "ORAFMTO Sample Formatting Program" on page B-7 for more information.

The Assembler copy file, ORASMFO, contains DSECTS that map and document the gateway SMF record fields. The ORASMFO data set member resides in the gateway SRCLIB library.

The ORASMF file is divided into the following sections:

- The standard record header section, which contains offsets and lengths of the Net and accounting sections
- The correlation section
- The OSDI section
- The database engine section
- The z/OS accounting section
- The Net section (if applicable), which contains information about the network origin of clients on an Oracle Net for z/OS TCP/IP protocol network (IBM or SNS/TCPaccess)
- The z/OS accounting section (if applicable)

Not all sections are present in all SMF records. For example, the z/OS accounting section is present only in SMF records for batch and TSO users. When a section is present, the SMF record header contains the correct length for that section, which might be release dependent, and the count field, which contains 1. The length field for nonexistent sections contains 0.

Contents of the SMF Header Section

Table C-1 contains brief descriptions for the labels in the SMF header section. For a complete layout of the contents of the SMF header section, refer to the DSECT.

Table B–1 Contents of the SMF Header Section

ORASMF0 Label	Description
SMFHDR	Standard SMF header
SMFHLEN	Total length of the SMF record
SMFHSEG	Segment descriptor = 0
SMFHSIN	SYS IND = X'80' Subsystem information to follow SYS IND = x'40' Subtype format record
SMFHREC	Record type default = 204 (decimal)
SMFHTIM	Timestamp, time (0.01 seconds since midnight)

Table B–1 (Cont.) Contents of the SMF Header Section

ORASMF0 Label	Description
SMFHDAT	Timestamp, date (0cyyydddf) c=0 for 19xx, c=1 for 20xx
SMFHSYS	System id
SMFHSSI	OSDI Subsystem id
SMFHSUB	Record subtype; 1 = accounting record
SMFSRVC	OSDI service name
SMFSESID	OSDI session id
SMFHRV1	Reserved
SMFNETO	Offset to Net section
SMFACTO	Offset to z/OS accounting section
SMFHRV2	Reserved
SMFNETL	Length of Net section
SMFACTL	Length of z/OS accounting section
SMFHRV3	Reserved

Contents of the SMF Correlation Section

Table B–2 contains brief descriptions for the labels in the SMF correlation section.

Table B–2 Contents of the SMF Correlation Section

ORASMF Label	Description
SMFAUTH	Authorization id = TSO logon id Batch userid on jobcard CICS USERID, TERM-ID,TRANS-ID, PROGRAM-ID, or OPID
SMFCORI	Correlation id = for TSO, logon id for batch, jobname for CICS, jobname Not valid for Oracle Net
SMFCONN	Connection type (TSO,BATCH,CICS,VTAM,TCP/IP,IMS)
SMFASID	Users address space id (not valid for Oracle Net)
SMFOUSR	Gateway logon id

Table B–2 (Cont.) Contents of the SMF Correlation Section

ORASMF Label	Description
SMFTNAME	Originating terminal id (if available)
SMFPNAME	Originating program name (if available)
SMFGRPN	RACF group name (if available)
SMFJBID	JES job identifier
SMFENTRY	RDR jobcard entry date (batch and TSO only). This field is equivalent to the SMF5RST field in the SMF job termination (type 5) record.
SMFEDATE	RDR jobcard entry date (batch and TSO only). This field is equivalent to the SMF5RSD field in the SMF job termination (type 5) record.

Contents of the SMF OSDI Data Section

Table B–3 Contents of the SMF OSDI Data Section

ORASMF0 Label	Description
SMFTIM	Beginning timestamp, time (0.01 second since midnight)
SMFDAT	Beginning timestamp, date (00yydddf), ending time and date in header
SMFDTAI	Data in
SMFDTAO	Data out
SMFXMCPU	Cross memory CPU time (TOD format)
SMFRPCS	RPC count
SMFHWST	High-water mark of storage used
SMFINV	Reason for invocation
SMFNORM	Normal termination
SMFabort	Clean up done

Contents of the SMF Database Engine Data Section

The database engine section is not applicable for Oracle Transparent Gateway for DB2.

Table B–4 Contents of the Database Engine Section

ORASMF Label	Description
SMFLRC	Logical read count
SMFPRC	Physical read count
SMFLWC	Logical writes
SMFDMC	DML COMMITs
SMFDMR	DML ROLLBACKs
SMFDED	DEADLOCKs
SMFHDLN	Length of SMF header

Contents of the SMF Oracle Net Data Section

Table B–5 Contents of the SMF Oracle Net Data Section

ORASMF Label	Description
SMFNET	Oracle Net section header
SMFNETL	Length of Net NIV information. The information contained in this section is specific to the Oracle Net driver in use. This information is variable length.
SMFNETA	Start of variable-length information

ORAFMTO Sample Formatting Program

A sample program, ORAFMTO, is provided with the gateway SMF interface to format gateway SMF records. ORAFMTO is an Assembler program that reads and formats SMF accounting records with the default gateway type of 199. It reads records from a variable-blocked sequential data set and writes the formatted records to a fixed block sequential data set with a logical record length of 132.

The sample ORAFMTO program is in the gateway SRCLIB library. The following members are included:

Member	Description
ORAFMTCL	contains sample JCL to compile and link ORAFMTO.
ORAFMTGO	contains sample JCL to run ORAFMTO.

ORAFMTO extracts and prints the following values from the gateway SMF records:

Table B–6 SMF Record Values

Value	Description
SSN	OSDI Subsystem name
SERVICE	OSDI Service name
SMFAUTH	Oracle userid assigned to the session using the database link.
SMFCONN	Connection type (TSO, BATCH, VTAM, TCP/IP)
ORACLE ID	User id assigned to the database link when the link is created. If the dblink is created without an associated userid, it defaults to the Oracle userid.
DATE	Start date of Oracle session
TIME	Start time of Oracle session
CPU SECONDS	Total CPU seconds used in the gateway address space. This number does not include time spent in the Oracle server address space, nor in the DB2 address space.
LOG READS	Not applicable to the gateway
PHY READS	Not applicable to the gateway
LOG WRITES	Not applicable to the gateway
DMC	Not applicable to the gateway
DMR	Not applicable to the gateway
DED	Not applicable to the gateway
HI STG	High-water mark of main storage used by the session

If the values are too large for the precision of their column, then they are shown as a series of asterisks.

Sample output from the ORAFMTO program:

```
SSN SMFAUTH SMFCONN ORACLE ID  DATE   TIME    TOTAL CPU SEC LOG READS LOG WRITES DMC DMR DDC DDR DED
--- ------- ------- -------- ------ ---------- ------------ --------- ---------- --- --- --- --- ---
ORA1 SSMITH BATCH   SCOTT    95.070 13:46:17.74       .755       152         23    6   2   1   1   0
ORA1 SSMITH BATCH   SCOTT    95.070 13:47:34.41       .193        88         23    6   2   1   1   0
ORA1 TJONES BATCH   SYSTEM   95.070 13:48:06.27       .269        95         23    6   2   1   1   0
ORA1 SSMITH BATCH   SCOTT    95.070 13:49:08.98       .084        36         10    3   2   0   0   0
ORA1 SSMITH BATCH   SCOTT    95.070 13:50:19.60       .120        21          2    1   0   0   0   0
ORA1        BATCH   SCOTT    95.070 13:52:04.67       .105        10         16    2   2   0   0   0
```

```
ORA1        BATCH   SCOTT    95.070 13:52:36.61      1.028       115         9    3   1   0   0   0
ORA1        BATCH   SCOTT    95.070 13:52:42.47       .576        71        23    5   3   1   1   0
ORA1        BATCH   SYSTEM   95.070  13:54:14.83      .110        10        16    2   2   0   0   0
```

C

Data Dictionary Views

This appendix includes the gateway data dictionary views accessible to all users of an Oracle database server. Most views can be accessed by any user with `SELECT` privileges for DB2 catalog tables.

This appendix include the following data dictionary views:

If a dictionary item is described with N/A in the following lists, then it means that the item is not available for the Oracle Transparent Gateway for DB2.

ALL_CATALOG

All tables:

`OWNER` is the owner of the object.

`TABLE_NAME` is the name of the object.

`TABLE_TYPE` is the type of object.

ALL_COL_COMMENTS

Comments on columns of all tables:

`OWNER` is the owner of the object.

`TABLE_NAME` is the object name.

`COLUMN_NAME` is the column name.

`COMMENTS` are the comments on the column.

ALL_CON_COLUMNS

Information about accessible columns in constraint definitions:

OWNER is the owner of the constraint definition.

CONSTRAINT_NAME is the name associated with the constraint definition.

TABLE_NAME is the name associated with the table containing the constraint definition.

COLUMN_NAME is the name associated with the column specified in the constraint definition.

POSITION is the original position of the column in the definition.

ALL_CONSTRAINTS

Constraint definitions on accessible tables:

OWNER is the owner of the constraint definition.

CONSTRAINT_NAME is the name associated with the constraint definition.

CONSTRAINT_TYPE is the type of constraint definition.

TABLE_NAME is the name associated with the table containing the constraint definition.

SEARCH_CONDITION is the text of the search condition for the table check.

R_OWNER is the owner of the table used in the referential constraint.

R_CONSTRAINT_NAME is the name of the unique constraint definition for the referenced table.

DELETE_RULE is the delete rule for the referential constraint.

STATUS is the status of the constraint.

ALL_IND_COLUMNS

Columns of the indexes on the accessible tables:

INDEX_OWNER is the owner of the index.

INDEX_NAME is the name of the index.

TABLE_OWNER is the table or cluster owner.

TABLE_NAME is the table or cluster name.

COLUMN_NAME is the column name.

COLUMN_POSITION is the position of column within index.

COLUMN_LENGTH is the indexed length of column.

ALL_INDEXES

Description of indexes on all tables:

OWNER is the owner of the index.

INDEX_NAME is the name of the index.

TABLE_OWNER is the owner of the indexed object.

TABLE_NAME is the name of the indexed object.

TABLE_TYPE is the type of the indexed object.

UNIQUENESS is the uniqueness status of the index.

TABLESPACE_NAME is the name of the tablespace containing the index.

INI_TRANS N/A

MAX_TRANS N/A

INITIAL_EXTENT N/A

NEXT_EXTENT N/A

MIN_EXTENTS N/A

MAX_EXTENTS N/A

PCT_INCREASE N/A

PCT_FREE N/A

BLEVEL is the depth of the index from its root block to its leaf blocks. A depth of one indicates that the root block and the leaf block are the same.

LEAF_BLOCKS is the number of leaf blocks in the index.

DISTINCT_KEYS is the number of distinct indexed values. For indexes enforcing UNIQUE and PRIMARY KEY constraints, this value is the same as the number of rows in the table.

AVG_LEAF_BLOCKS_PE N/A

AVG_DATA_BLOCKS_PE N/A

CLUSTERING_FACTOR N/A

STATUS is the valid state of the index: VALID.

ALL_OBJECTS

All indexes and tables:

OWNER is the owner of the object.

OBJECT_NAME is the name of object.

OBJECT_ID is the object number of the object.

OBJECT_TYPE is the type of object.

CREATED N/A

LAST_DDL_TIME N/A

TIMESTAMP N/A

STATUS is the state of the object.

ALL_SYNONYMS

All synonyms accessible to the user:

OWNER is the owner of the synonym.

SYNONYM_NAME is the name of the synonym.

TABLE_OWNER is the owner of the object referenced by the synonym.

TABLE_NAME is the name of the object referenced by the synonym.

DB_LINK N/A

ALL_TAB_COLUMNS

Columns of all tables:

OWNER is the owner of the table or view.

TABLE_NAME is the table or view name.

COLUMN_NAME is the column name.

DATA_TYPE is the data type of the column.

DATA_LENGTH is the maximum length of the column in bytes.

DATA_PRECISION N/A

DATA_SCALE are the digits to the right of the decimal point in a number.

NULLABLE asks if the column allows nulls. The value is n if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY key.

COLUMN_ID is the sequence number of the column that is created.

DEFAULT_LENGTH N/A

DATA_DEFAULT N/A

NUM_DISTINCT is the number of distinct values in each column of the table.

LOW_VALUE and HIGH_VALUE are the second lowest and second highest values for tables. These statistics are expressed in hexadecimal notation for the internal representation of the first 32 bytes of the values.

DENSITY N/A

ALL_TAB_COMMENTS

Comments on all tables:

OWNER is the owner of the object.

TABLE_NAME is the name of the object.

TABLE_TYPE is the type of object.

COMMENTS are comments on the object.

ALL_TABLES

Description of all tables:

OWNER is the owner of the table.

TABLE_NAME is the name of the table.

TABLESPACE_NAME is the name of the tablespace containing the table.

CLUSTER_NAME N/A

PCT_FREE N/A

PCT_USED N/A

INI_TRANS N/A

MAX_TRANS N/A

INITIAL_EXTENT N/A

NEXT_EXTENT N/A

MIN_EXTENTS N/A

MAX_EXTENTS N/A

PCT_INCREASE N/A

BACKED_UP N/A

NUM_ROWS is the number of rows in the table.

BLOCKS N/A

EMPTY_BLOCKS N/A

AVG_SPACE N/A

CHAIN_CNT N/A

AVG_ROW_LEN is the average length of a row in the table in bytes.

ALL_USERS

Information about current users of the database:

USERNAME is the name of the user.

USER_ID N/A

CREATED N/A

ALL_VIEWS

Text of views accessible to the user:

OWNER is the owner of the view.

VIEW_NAME is the name of the view.

TEXT_LENGTH is the length of the view text.

TEXT is the view text. Only the first row of text is returned, even if multiple rows exist.

COLUMN_PRIVILEGES

Grants on columns for which the user is the grantor, grantee or owner, or PUBLIC is the grantee:

GRANTEE is the name of the user to whom access was granted.

OWNER is the user name of the object's owner.

TABLE_NAME is the name of the object.

COLUMN_NAME is the name of the column.

GRANTOR is the name of the user who performed the grant.

INSERT_PRIV is the permission to insert into the column.

UPDATE_PRIV is the permission to update the column.

REFERENCES_PRIV is the permission to reference the column.

CREATED is the time stamp for the grant.

OTGDB2.OTGREGISTER

DB2 special registers:

CURRENT_SQLID is the DB2 SQL authorization id.

CURRENT_USER is the DB2 primary authorization id.

CURRENT_DATE is the DB2 current date.

CURRENT_TIME is the DB2 current time.

CURRENT_TIMESTAMP is the DB2 current time stamp.

CURRENT_TIMEZONE is the DB2 current time zone.

CURRENT_SERVER is the DB2 location name of current server.

CURRENT_PACKAGESET is the DB2 collection id of the package used for running SQL statements.

CURRENT_LC_CTYPE is the DB2 locale for functions.

CURRENT_OPTIM_HINT is the DB2 CURRENT OPTIMIZATION HINT special register - optimization hint.

CURRENT_PATH is the DB2 search path for unqualified functions and procedures.

CURRENT_PRECISION is the DB2 CURRENT PRECISION special register - numeric precision.

CURRENT_RULES is the DB2 CURRENT RULES special register - SQL rules in effect.

TABLE_PRIVILEGES

Grants on objects for which the user is the grantor, grantee or owner, or PUBLIC is the grantee:

GRANTEE is the name of the user to whom access is granted.

OWNER is the owner of the object.

TABLE_NAME is the name of the object.

GRANTOR is the name of the user who performed the grant.

SELECT_PRIV is the permission to select from an object.

INSERT_PRIV is the permission to insert into an object.

DELETE_PRIV is the permission to delete from an object.

UPDATE_PRIV is the permission to update an object.

REFERENCES_PRIV N/A

ALTER_PRIV is the permission to alter an object.

INDEX_PRIV is the permission to create or drop an index on an object.

CREATED is the time stamp for the grant.

USER_CATALOG

Tables owned by the user:

TABLE_NAME is the name of the object.

TABLE_TYPE is the type of object.

USER_COL_COMMENTS

Comments on columns of user's tables:

TABLE_NAME is the object name.

COLUMN_NAME is the column name.

COMMENTS are the comments on the column.

USER_CONS_COLUMNS

Information about columns in constraint definitions owned by the user:

OWNER is the owner of the constraint definition.

CONSTRAINT_NAME is the name associated with the constraint definition.

TABLE_NAME is the name associated with the table with the constraint definition.

COLUMN_NAME is the name associated with the column specified in the constraint definition.

POSITION is the original position of the column in the definition.

USER_CONSTRAINTS

Constraint definitions on user's tables.

OWNER is the owner of the constraint definition.

CONSTRAINT_NAME is the name associated with the constraint definition.

CONSTRAINT_TYPE is the type of constraint definition.

TABLE_NAME is the name associated with the table with the constraint definition.

SEARCH_CONDITION is the text of the search condition for the table check.

R_OWNER is the owner of the table used in the referential constraint.

R_CONSTRAINT_NAME is the name of the unique constraint definition for the referenced table.

DELETE_RULE is the delete rule for the referential constraint.

STATUS is the status of the constraint.

USER_INDEXES

Description of the user's own indexes:

INDEX_NAME is the name of the index.

TABLE_OWNER is the owner of the indexed object.

TABLE_NAME is the name of the indexed object.

TABLE_TYPE is the type of the indexed object.

UNIQUENESS is the uniqueness status of the index.

TABLESPACE_NAME is the name of the tablespace containing the index.

INIT_TRANS N/A

MAX_TRANS N/A

INITIAL_EXTENT N/A

NEXT_EXTENT N/A

MIN_EXTENTS N/A

MAX_EXTENTS N/A

PCT_INCREASE N/A

PCT_FREE N/A

BLEVEL is the depth of the index from its root block to its leaf blocks. A depth of one indicates that the root block and the leaf block are the same.

LEAF_BLOCKS is the number of leaf blocks in the index.

DISTINCT_KEYS is the number of distinct indexed values. For indexes enforcing UNIQUE and PRIMARY KEY constraints, this value is the same as the number of rows in the tables.

AVG_LEAF_BLOCKS_PE N/A

AVG_DATA_BLOCKS_PE N/A

CLUSTERING_FACTOR N/A

STATUS is the state if the indexes: VALID.

USER_OBJECTS

Objects owned by the user:

OWNER is the owner of the object.

OBJECT_NAME is the name of the object.

OBJECT_ID is the object number of the object.

OBJECT_TYPE is the type of object.

CREATED N/A

LAST_DDL_TIME N/A

TIMESTAMP N/A

STATUS is the state of the object: VALID.

USER_SYNONYMS

Private synonyms owned by the user:

SYNONYM_NAME is the name of the synonym.

TABLE_OWNER is the owner of the object referenced by the synonym.

TABLE_NAME is the name of the object referenced by the synonym.

DB_LINK N/A

USER_TAB_COLUMNS

Columns of user's tables:

TABLE_NAME is the table, view, or cluster name.

COLUMN_NAME is the column name.

DATA_TYPE is the data type of the column.

DATA_LENGTH is the maximum length of the column in bytes.

DATA_PRECISION N/A

DATA_SCALE are the digits to the right of the decimal point in a number.

NULLABLE asks if the column allow nulls. The value is n if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY key.

COLUMN_ID is the sequence number of the column that is created.

DEFAULT_LENGTH N/A

DATA_DEFAULT N/A

NUM_DISTINCT is the number of distinct values in each column of the table.

LOW_VALUE and HIGH_VALUE are the second lowest and the second highest values respectively, for tables with more than three rows. These statistics are expressed in hexadecimal notation for the internal representation of the first 32 bytes of the values.

DENSITY N/A

USER_TAB_COMMENTS

Comments on the tables owned by the user:

TABLE_NAME is the name of the object.

TABLE_TYPE is the type of object.

COMMENTS are the comments on the object.

USER_TABLES

Description of the user's own tables:

TABLE_NAME is the name of the table.

TABLESPACE_NAME is the name of the tablespace containing the table.

CLUSTER_NAME N/A

PCT_FREE N/A

PCT_USED N/A

INI_TRANS N/A

MAX_TRANS N/A

INITIAL_EXTENT N/A

NEXT_EXTENT N/A

MIN_EXTENTS N/A

MAX_EXTENTS N/A

PCT_INCREASE N/A

BACKED_UP N/A

NUM_ROWS is the number of rows in the table.

BLOCKS N/A

EMPTY_BLOCKS N/A

AVG_SPACE N/A

CHAIN_CNT N/A

AVG_ROW_LEN is the average length of a row in the table in bytes.

USER_USERS

Information about the current user:

USERNAME is the name of the user.

USER_ID N/A

DEFAULT_TABLESPACE N/A

TEMP_TABLESPACE N/A

CREATED N/A

USER_VIEWS

Text of views owned by the user:

VIEW_NAME is the name of the view.

TEXT_LENGTH is the length of the view text.

TEXT is the first line of view text.

D

Quick Reference to Oracle SQL Functions

This appendix lists the Oracle SQL functions: ABS, ACOS, ADD_MONTHS, ASIN, ASCII, ATAN, ATAN2, CEIL, CHAR_TO_ROWID, CHR, CONVERT, COS, COSH, DECODE, DUMP, EXP, FLOOR, GREATEST, HEXTORAW, INITCAP, INSTR, INSTRB, LAST_DAY, LEAST, LENGTH, LENGTHB, LN, LOG, LOWER, LPAD, LTRIM, MOD, MONTHS_BETWEEN, NEW_TIME, NEXT_DAY, NLS_INITCAP, NLS_LOWER, NLS_UPPER, NLSSORT, POWER, RAWTOHEX, REPLACE, ROUND, ROWIDTOCHAR, RPAD, RTRIM, SIGN, SIN, SINH, SOUNDEX, SQRT, STDDEV, SUBSTR, SUBSTRB, SYSDATE, TAN, TANH, TO_CHAR, TO_DATE, TO_LABEL, TO_MULTI_BYTE, TO_NUMBER, TO_SINGLE_BYTE, TRANSLATE, TRUNC, UID, UPPER, USER, USERENV, VARIANCE, and VSIZE.

E

Sample Applications

This appendix contains sample applications that can be used with the gateway.

This appendix includes the following examples:

DB2IND

DB2IND is a sample DB2 stored procedure that inserts a row into a DB2 table. This procedure uses the SIMPLE WITH NULLS linkage convention.

```
/*****************************************************************/
/* This DB2 stored procedure uses indicator variables to insert null */
/* values for DNAME and LOC columns of DB2 user table SCOTT.DEPT.    */
/* SCOTT.DEPT table is defined to DB2 as DEPTNO INTEGER, DNAME       */
/* CHAR(14), LOC VARCHAR(13).  This procedure receives 3 input       */
/* parameters from the calling program which contain the values to   */
/* insert for DEPTNO, DNAME, and LOC.                                */
/*                                                                   */
/* The linkage convention used for this stored procedure is SIMPLE   */
/* WITH NULLS.                                                       */
/*                                                                   */
/* The output parameter for this procedure contains the SQLCODE from */
/* the INSERT operation.                                             */
/*                                                                   */
/* The entry in the DB2 catalog table SYSIBM.SYSPROCEDURES for this  */
/* stored procedure might look like this:                            */
/*                                                                   */
/* INSERT INTO SYSIBM.SYSPROCEDURES                                  */
/*   (PROCEDURE, AUTHID, LUNAME, LOADMOD, LINKAGE, COLLID, LANGUAGE, */
```

```
/*    ASUTIME, STAYRESIDENT, IBMREQD, RUNOPTS, PARMLIST)            */
/* VALUES                                                           */
/*   ('DB2IND', ' ', ' ', 'DB2IND', 'N', 'DB2DEV', 'C', '0', ' ',   */
/*    'N', ' ', 'A INT IN, B CHAR(14) IN, C VARCHAR(13) IN,         */
/*    D INT OUT, E CHAR(10) OUT');                                  */
/*******************************************************************/
#pragma runopts(plist(os))
#include <stdlib.h>
  EXEC SQL INCLUDE SQLCA;
/*******************************************************************/
/* Declare C variables for SQL operations on the parameters.  These */
/* are local variables to the C program which you must copy to and  */
/* from the parameter list provided to the stored procedure.        */
/*******************************************************************/
  EXEC SQL BEGIN DECLARE SECTION;
  long dno;               /* input parm - DEPTNO */
  char dname[15];         /* input parm - DNAME  */
  char locale[14];        /* input parm - LOC    */
  struct INDICATORS {
     short int   i1;
     short int   i2;
     short int   i3;
     short int   o;
  } indvar;               /* indicator variable structure */
  EXEC SQL END DECLARE SECTION;
  main(argc,argv)
    int argc;
    char *argv[];
   {
/*******************************************************************/
/* Copy the input parameters into the area reserved in the local    */
/* program for SQL processing.                                      */
/*******************************************************************/
    dno = *(int *) argv[1];
    strcpy(dname, argv[2]);
    strcpy(locale, argv[3]);
/*******************************************************************/
/* Copy indicator variable values for the parameter list.           */
/*******************************************************************/
    memcpy(&indvar,(struct INDICATORS *) argv[6], sizeof(indvar));
/*******************************************************************/
/* Issue SQL INSERT to insert a row into SCOTT.DEPT                 */
/*******************************************************************/
 EXEC SQL INSERT INTO SCOTT.DEPT VALUES
(:dno:indvar.i1, :dname:indvar.i2, :locale:indvar.i3);
```

```
/*********************************************************************/
/* Copy SQLCODE to the output parameter list.                        */
/*********************************************************************/
    *(int *) argv[4] = SQLCODE;
    indvar.o = 0;
/*********************************************************************/
/* Copy indicator variable values back to the output parameter list.  */
/*********************************************************************/
    memcpy( (struct INDICATORS *) argv[6], &indvar, sizeof(indvar));
   }
```

ORAIND

ORAIND is a sample host program that calls a DB2 stored procedure (DB2IND) to insert a row into a DB2 table. Embedded PL/SQL is used to manipulate the indictor variables.

```
/*********************************************************************/
/* This sample ProC program calls DB2 stored procedure DB2IND to     */
/* insert null values into DB2 user table SCOTT.DEPT.  This calling  */
/* program uses embedded PL/SQL to pass indicator variables in the   */
/* parameter list of the DB2 stored procedure call.                  */
/*********************************************************************/
#include <stdio.h>
EXEC SQL BEGIN DECLARE SECTION;
        VARCHAR         username[20];
        VARCHAR         password[20];
        int             dept_no;
        char            dept_name[14];
        VARCHAR         location[13];
        int             code;
        char            buf[11];
        short           ind1;
        short           ind2;
        short           ind3;
        short           oind;
        int             x;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
main()
{
/*********************************************************************/
/* Setup Oracle userid and password                                  */
/*********************************************************************/
```

```
  strcpy(username.arr, "SCOTT");          /* copy the username */
  username.len = strlen(username.arr);
  strcpy(password.arr, "TIGER");          /* copy the password */
  password.len = strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR GOTO sqlerror;
/*****************************************************************/
/* Logon to Oracle                                               */
/*****************************************************************/
  EXEC SQL CONNECT :username IDENTIFIED BY :password;
  printf("\nConnected to ORACLE as user: %s\n", username.arr);
  /* Delete any existing rows from DB2 table */
  EXEC SQL DELETE from SCOTT.DEPT@GTWLINK where LOC='INDVARS';
  EXEC SQL COMMIT;
/*----------------------- begin pl/sql block ----------------------*/
/*****************************************************************/
/* Insert 5 rows into DB2 table SCOTT.DEPT by invoking DB2 stored    */
/* procedure DB2IND.  Use indicator variables to pass null values to */
/* the stored procedure.  The DB2 stored procedure will perform the  */
/* INSERT.                                                           */
/*                                                                   */
/* SCOTT.DEPT table is defined on DB2 as:                            */
/*                                                                   */
/*    DEPTNO    INTEGER;                                           */
/*    DNAME     CHAR(14);                                          */
/*    LOC       VARCHAR(13);                                       */
/*                                                                   */
/*****************************************************************/
  EXEC SQL EXECUTE
  DECLARE
    buf   char(10);
  BEGIN
  for i in 1 ..  5 loop
     :dept_no:ind1 := 10 * i;
     :dept_name:ind2 := null;
     :location:ind3 := null;
     SYSPROC.DB2IND@GTWLINK
     (:dept_no:ind1, :dept_name:ind2, :location:ind3, :code:oind, buf);
  end loop;
  END;
  END-EXEC;
/*------------------------ end pl/sql block -----------------------*/
/*****************************************************************/
/* Verify row insertion.  Use indicator variables to check columns   */
/* for null values.  Update the column with a value if column is     */
/* null.                                                             */
```

```
/*************************************************************************/
  for (x = 10; x < 60; x = x + 10)
  {
     EXEC SQL SELECT deptno, dname, loc into
        :dept_no:ind1, :dept_name:ind2, :location:ind3
        from SCOTT.DEPT@GTWLINK where deptno = :x;
     if ((ind2 == -1) && (ind3 == -1))
     {
        printf("\nAfter INSERT\n");
        printf("\ndeptno = %d, dname = NULL, loc = NULL\n", dept_no);
        EXEC SQL UPDATE SCOTT.DEPT@GTWLINK set dname = 'TESTING',
           loc = 'INDVARS' where deptno = :x;
        EXEC SQL COMMIT;
     }
     EXEC SQL SELECT deptno, dname, loc into
        :dept_no:ind1, :dept_name:ind2, :location:ind3
        from SCOTT.DEPT@GTWLINK where deptno = :x;
     printf("\nAfter UPDATE:\n");
     printf("\ndeptno = %d, dname = %s, loc = %s\n",
        dept_no, dept_name, location.arr);
  }
/*************************************************************************/
/* Logoff from Oracle                                                    */
/*************************************************************************/
  EXEC SQL COMMIT RELEASE;
  printf("\n\nHave a good day\n\n");
  exit(0);
  sqlerror:
    printf("\n% .70s \n", sqlca.sqlerrm.sqlerrmc);
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL ROLLBACK RELEASE;
    exit(1);
}
```

F

Installation Reference

This appendix documents additional installation information that is referenced in the installation sections. This appendix includes the following section:

Choosing Data Set Name Qualifiers

The gateway installation setup and initialization process creates the first of several z/OS data sets. Later in the installation, you can specify the high-level and second-level data set name qualifiers that are used for subsequently created data sets. Refer to the *Oracle Database Installation Guide for IBM z/OS (OS/390)* for a list the data sets and their default space allocations.

Oracle recommends that you use the same qualifiers for all of the installation-related data sets. At this time, you need to choose and use the qualifiers that were selected during installation setup and initialization.

While choosing qualifiers, remember the following requirements:

- You must choose unique qualifiers.

  Using different qualifiers ensures that the products in the product set are maintained in separate libraries as required.

Attention:

1. Do not use the same qualifiers that you have used for any other Oracle for z/OS product set that you have previously installed. If you do, then the installation procedures will delete and reallocate your current Oracle libraries.

2. Do not concatenate these libraries with any existing libraries that you are running for previously installed product sets.

- In most z/OS systems, some preparation is required before creating data sets with a new high-level data set name qualifier.

  If you intend to use a new high-level qualifier for your Oracle data sets, then you must define an `ALIAS` before running the job that loads the installation JCL. When in doubt, ask your z/OS systems programmer for assistance.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z