Oracle® Database Gateway for DRDA

User's Guide

11*g* Release 1 (11.1)

B31046-01

June 2007

ORACLE®

Oracle Database Gateway for DRDA User's Guide, 11*g* Release 1 (11.1)

B31046-01

Primary Author: Maitreyee Chaliha

Contributing Author: Denis Raphaely, Peter A. Castro

Contents

2 Release Information

3 Using the Oracle Database Gateway for DRDA

4 Developing Applications

5 Error Messages, Diagnosis, and Reporting

A Oracle DB2 Data Dictionary Views

B Initialization Parameters

Preface

The Oracle Database Gateway for DRDA provides users with transparent access to DRDA databases as if they were Oracle databases.

Intended Audience

This guide is intended for anyone responsible for installing, configuring, and administering the gateway, and also for application developers.

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

- Installing and configuring the Oracle Database Gateway for DRDA
- Configuring TCP/IP
- Setting up gateway security
- Diagnosing gateway errors
- Using the gateway to access tables in DRDA databases
- Writing applications that access DRDA databases through the gateway

You must understand the fundamentals of Oracle Database Gateway and the operating system you are working on before using this guide to install or administer the gateway.

Processors

Refer to the *Oracle Database Installation Guide* and to the certification matrix on Oracle Meta*Link* for the most up-to-date list of certified hardware platforms and operating system version requirements to operate the gateway for your Linux, AIX-Based, HP-UX, or Solaris system. The Oracle Meta*Link* web site can be found at the following URL:

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

The gateway processor requirements for your platform are as follows:

- for Linux 32-bit: Intel Pentium-based processors
- for Linux 64-bit: Intel Itanium 2 or AMD64 based 64-bit systems
- for Linux S/390: Any processor that can run Linux S/390
- for AIX-Based Systems: IBM pSeries
- for HP-UX: HP 9000 Series HP-UX that can run the required version of HP-UX

- for Solaris: A Solaris Operating System (SPARC 64-bit) that can run the required version of Solaris with 64-bit architecture

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.

Related Documents

The *Oracle Database Gateway for DRDA User's Guide* is included as part of your product shipment. Also included is:

Oracle Database Heterogeneous Connectivity Administrator's Guide

This guide contains information common to all heterogeneous gateways, including important information on functions, parameters, and error messages.

Oracle Database Administrator's Guide

Oracle Database Concepts

Oracle Database Error Messages

Oracle Database Performance Tuning Guide

Oracle Database Security Guide

Oracle Database Heterogeneous Connectivity Administrator's Guide

Typographic Conventions

The following typographic conventions are used in this guide:

Convention	Description
`monospace`	Monospace type indicates commands, directory names, user names, path names, and file names.
italics	Italic type indicates variables, including variable portions of file names. It is also used for emphasis and for book titles.
UPPERCASE	Uppercase letters indicate Structured Query Language (SQL) reserved words, initialization parameters, and environment variables.
Bold	Bold type indicates screen names and fields.
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.

SQL*Plus Prompts

The SQL*Plus prompt, `SQL>`, appears in SQL statements 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:

- KB, for kilobyte, which equals 1,024 bytes
- MB, for megabyte, which equals 1,048,576 bytes
- GB, for gigabyte, which equals 1,073,741,824 bytes

1 Introduction

The Oracle Database Gateway for DRDA enables you to:

- Integrate heterogeneous database management systems so that they appear as a single homogeneous database system
- Read and write data from Oracle applications to data in DB2/OS390, DB2/400, and DB2 Universal Database databases in addition to any Oracle database data.

This chapter provides information about the architecture, uses, and features of the Oracle Database Gateway for DRDA.

This chapter contains the following sections:

Introduction to the Oracle Database Gateway

In todays global economy, information is the most valuable resource of a company. Whether you need to analyze new markets, tailor your products to meet local demands, increase your ability to handle complex customer information, or streamline operations, your company requires instant access to current and complete information.

Company growth and diversification often mean functioning with a collage of applications and geographically scattered data that may be using incompatible networks, platforms, and storage formats. Diverse application standards and storage formats can make integration of information difficult. Oracle offers integration technologies to overcome these technical barriers. Oracle Open Gateways simplify complex systems and remove obstacles to information, providing your company the opportunity to focus on business.

Protection of Current Investment

Oracle Database Gateway for DRDA gives your company the ability to develop its information systems without forfeiting its investments in current data and applications. The gateway gives you access to your Oracle data and DB2 data with a single set of applications while you continue to use existing IBM applications to access your DB2 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 choose to migrate to Oracle database technology, then the gateway enables you to control the pace of your migration. As you transfer applications from your previous technology to the Oracle database, you can use the gateway to move the DB2 data into Oracle databases.

Release 11*g* Gateways

The Oracle Database 11*g* provides the foundation for the next generation of the Oracle Open gateways Release 11g, which will deliver enhanced integration capabilities by exploiting Oracle Database 11*g* Heterogeneous Services. Heterogeneous Services is a component of Oracle Database 11*g*. Oracle Database 11*g* provides the common architecture for future generations of the gateways. For detailed information on Oracle Heterogeneous Services, refer to Oracle Database Heterogeneous Connectivity Administrator's Guide.

The version 10 gateways are even more tightly integrated with Oracle Database 11*g* than previous versions, 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 11*g*, reducing the number of round trips and the amount of data sent over the network. SQL execution 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 any enhancements in Oracle Database 11*g*, and you can quickly extend those benefits to your non-Oracle data.

Advantages of Oracle Database Gateway for DRDA

Oracle Database Gateway for DRDA enables Oracle applications to access the DRDA Application Servers, such as DB2 for MVS, through SQL. The gateway and Oracle Database 11*g* together create the appearance that all data resides on a local Oracle Database 11*g*, even though data might be widely distributed. If data is moved from a DRDA Application Server database to an Database server, then no changes in application design or function are needed. The gateway handles all differences in both data types and SQL functions between the application and the database.

Gateway Capabilities

Oracle Database Gateway for DRDA gives you the power to integrate your heterogeneous system into a single, seamless environment. This integration enables 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 you can avoid the tedious, error-prone process of manual data transfer. Together with the Oracle tools, networking, and data server technology, the Oracle Database Gateway for DRDA sets a high standard for seamless, enterprise-wide information access.

Oracle Database Gateway for DRDA enables applications to read and update DB2 data and Oracle data as if all of the data were stored in a single database. As a result, end

users and application programmers are not required to know either the physical location or the storage characteristics of the data. This transparency not only enables you to integrate heterogeneous data seamlessly, it simplifies your gateway implementation, application development, and maintenance.The gateway capabilities are as follows:

- Transparency at All Levels
- Extended Database Services
- Extended Advanced Networking, Internet and Intranet Support
- Dynamic Dictionary Mapping
- SQL
- Data Definition Language
- Data Control Language
- Passthrough and Native DB2 SQL
- Stored Procedures
- Languages
- Oracle Database Technology and Tools
- SQL*Plus
- Two-Phase Commit and Multi-Site Transactions
- Site Autonomy
- Migration and Coexistence
- Security

Transparency at All Levels

The Oracle Database Gateway for DRDA gives you transparency at every level within your enterprise. Oracle Database Gateway gives you transparency at the following levels:

- Location

  End users can access tables by name without needing to understand the physical location of the tables.

- Network

  The gateways exploit the Oracle Net technology to allow users to access data across multiple networks without concern for the network architecture. TCP/IP protocol is supported.

- Operating system

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

- Data storage

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

- Access method

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

Extended Database Services

Following are some of the sophisticated Oracle Database 11*g* services available through the gateway:

- SQL functions

  Your application can access all your data using Oracle SQL, which is rich in features. Advanced Oracle Database 11*g* functions, such as outer joins, are available even if the target data stores do not support them in a native environment. The method by which the gateways are integrated with the Oracle Database 11*g* 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 11*g* can utilize its advanced query optimization techniques to ensure that SQL statements are executed efficiently against any of your data. The data distribution and storage characteristics of local and remote data are equally considered.

- Two-phase commit protection

  The database server 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 will be affected.

- Stored procedures and database triggers

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

Extended Advanced Networking, Internet and Intranet Support

The gateway integration with the Oracle Database 11*g* extends (to non-Oracle data) the benefits of the Oracle Internet and Oracle Net software and extends the benefits of 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 Oracle database can also incorporate information from data stores accessible through the gateways. Web browsers can connect to the Oracle database using any application server product that supports Oracle software.

- Implicit protocol conversion

  Oracle and Oracle Net can work together as a protocol converter, allowing applications to transparently access other data stores on platforms that do not support the clients network protocol. An Oracle Database 11*g* can use TCP/IP to communicate with the gateway and another data store.

- 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 Advanced Security.

- Wireless communication

  Oracle Mobile Agents, an Oracle industry-leading mobile technology, enables wireless communication to Oracle database or to any databases that are accessible through the gateways. This gives your 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 that information in a separate data dictionary stored in a hub. Applications then access the information using the hub dictionary instead of the native dictionaries of each database. This approach requires a great deal of manual configuration and maintenance on your part. As administrators, you must update the data dictionary in the hub whenever the structure of a remote table is changed.

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

SQL

Oracle Database Gateways ease your 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 issue native data control language (DCL) statements from an Oracle environment, allowing central administration of user privileges and access levels for heterogeneous data stores.

Passthrough and Native DB2 SQL

Execution of native DB2 SQL can be passed through the gateway for execution directly against DB2. This enables applications to send statements, such as a DB2 `CREATE TABLE`, to the gateway for execution 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 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 11*g*. Using Oracle stored procedures can increase your 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 execute DB2 stored procedures using standard Oracle PL/SQL. The Oracle application executes the DB2 stored procedure as if it were an Oracle remote procedure.

Languages

Any application or tool that supports the Oracle Database 11*g* can access over thirty different data sources through the Oracle gateways. A wide variety of open system tools from Oracle Corporation and third-party vendors can be used, even if the data is stored in legacy, proprietary formats. Hundreds of tools are supported, including ad hoc query tools, Web browsers, turnkey applications, and application development tools.

Oracle Database Technology and Tools

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

SQL*Plus

You can use SQL*Plus for moving data between databases. This product gives you the ability to copy data from your department databases to corporate Oracle database instances.

Two-Phase Commit and Multi-Site Transactions

The gateway can participate as a partner in multi-site 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:

- Full two-phase commit partner
- Commit point site
- Single-site update partner
- Read-only partner

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

Oracle Database Gateway for DRDA, 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. Refer to "Read-Only Gateway" on page 3-6 in Chapter 3, "Using the Oracle Database Gateway for DRDA".

Site Autonomy

All Oracle database products, including gateways, supply site autonomy. For example, administration of a data source remains the responsibility of the original system administrator. Site autonomy also functions such that gateway products do not override the security measures established by 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. The result is that the Oracle database 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.

Functionally, gateway security is identical to that of an Oracle database, as described in the *Oracle Database Administrator's Guide*. Oracle database security is mapped to the data dictionary of the data source.

Terms

The terms used in this guide do not necessarily conform to the IBM terminology. The following list presents several terms and their meanings as used within this guide:

DRDA data is, generically, any database data accessed through DRDA.

DRDA database is the collection of data that belongs to a DRDA server

DRDA server is a database server that can be accessed through DRDA. IBM terminology for a DRDA server is a DRDA Application Server, or AS.

DRDA server type is a specific database product or program that can act as a DRDA server.

Oracle database is any Oracle Database 11*g* instance that communicates with the Oracle Database Gateway for DRDA to distribute database access operations to a DRDA server. The Oracle database can also be used for non-gateway applications.

DB2 Universal Database is a generic name for the UNIX-based or Windows-based implementations of DB2. DB2/UDB is frequently used as an abbreviation for DB2 Universal Database.

Architecture

The Oracle Database Gateway for DRDA works with the Oracle Database 11*g*

to shield most of the differences of the non-Oracle database from Oracle applications. This means that the Oracle applications can access the Oracle Database 11*g* data and the DRDA database data as if it were Oracle database data located at the Oracle database.

The architecture consists of the following main components:

- Client

  The client is an Oracle application or tool.

- Oracle database

  The Oracle database instance is accessed by an Oracle Database 11*g* with procedural and distributed options. Usually, the Oracle database is installed on the same host as the gateway, but this is not a requirement. The Oracle database and the gateway communicate in the normal Oracle database-to-server manner.

  If the Oracle database is not on the host where the gateway resides, then you must install the correct Oracle networking software on the platform where the server resides. For Oracle Database 11*g*, you must install Oracle Net on the Oracle Database 11*g* machine.

- Oracle Database Gateway for DRDA

  The gateway must be installed on hosts that are running the appropriate operating system.

  If the Oracle database is not on the same host, then you must also install Oracle Net so that the gateway and Oracle Database 11*g* can communicate.

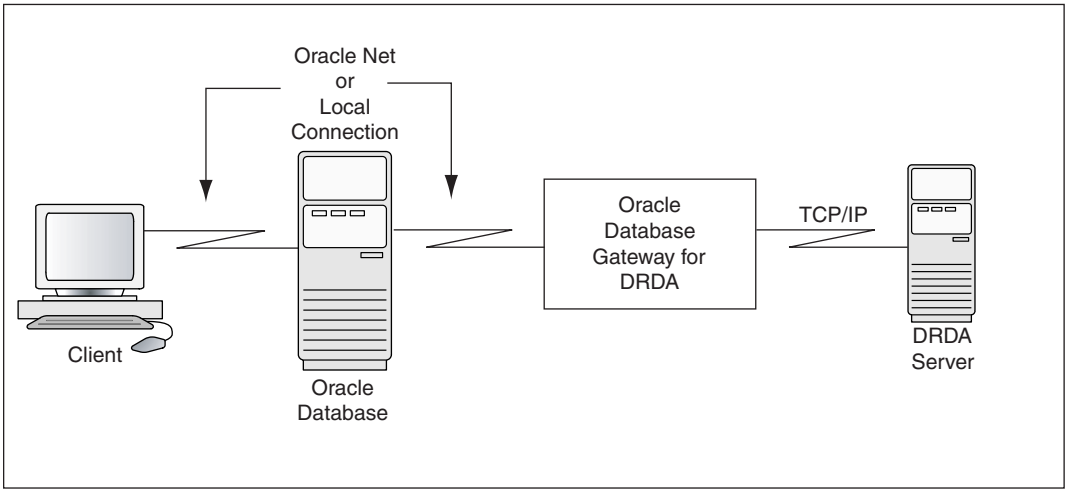
- DRDA server

  The DRDA server must be a DRDA server database on a system accessible to the host via a network.

Multiple Oracle Database 11*g* servers can access the same gateway. A single host gateway installation can be configured to access more than one DRDA server.

Figure 1–1 illustrates the gateway architecture.

Figure 1–1 The Gateway Architecture



Implementation

When the gateway is installed on your host, it has some of the same components as an Oracle database instance on your host. The gateway has the following components:

- A base file directory, similar to the one associated with an Oracle instances `ORACLE_HOME` environment variable

- A gateway system identifier (SID), comparable to an Oracle instances `ORACLE_SID`
- Oracle Net to support communication between the Oracle database and the Oracle Database Gateway for DRDA

The gateway does not have:

- Control, redo log, or database files
- The full set of subdirectories and ancillary files that are associated with an installed Oracle Database 11g

Because the gateway does not have background processes and does not need a management utility, such as Oracle Enterprise Manager, you do not need to start the gateway product. Each Oracle Database 11g user session that accesses a particular gateway creates an independent process on the host. This process runs the gateway session and executes network operations to communicate with a DRDA server.

How the Gateway Works

The gateway has no database functions of its own. Instead, it provides an interface by which an Oracle Database 11g can direct part or all of a SQL operation to a DRDA database.

The gateway that is supporting the DRDA server is identified to the Oracle database using a database link. The database link is the same construct that is used to identify other Oracle Database 11g servers. Tables on the DRDA 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, then you can refer to tables on the DRDA server by using simple names as though the table were local to the Oracle database.

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

The gateway is responsible for sending these SQL statements to the DRDA server for execution and for fielding and returning responses. The responses are either data or messages. Any conversions between Oracle data types and DRDA data types are performed by the gateway. Both the Oracle database and the application read and process only Oracle data types.

SQL Differences

Not all SQL implementations are the same. The Oracle Database 11*g* supports a larger set of built-in functions than the databases that are currently accessed through the gateway. The Oracle database and the gateway work together to convert SQL to a form that is compatible with the specific DRDA server.

During this conversion, an Oracle Database 11*g* function can be converted to a function that is recognizable to the specific DRDA server. For example, the Oracle Database 11*g* `NVL` function is converted to the DB2 `VALUE` function.

Alternatively, the Oracle database withholds functions that are not executable by the DRDA server and performs them after rows are fetched from the DRDA database. This processing generally applies to `SELECT` statements. The Oracle database and the gateway cannot perform this kind of manipulation on `UPDATE`, `INSERT`, or `DELETE` statements because doing so changes transaction semantics.

Oracle Tools and the Gateway

Use the Oracle Database Gateway to run applications, such as Oracle database tools, that read and write data that is stored in DRDA databases.

While the Oracle Database Gateway for DRDA provides no new application or development facilities, it extends the reach of existing Oracle database tools to include data in non-Oracle databases that support DRDA.

Using the Oracle Database Gateway for DRDA with other Oracle products can greatly extend the capabilities of the stand-alone gateway.

SQL*Plus

Use SQL*Plus and the Oracle Database Gateway for DRDA to create a distributed database system, providing an easy-to-use transfer facility for moving data between the distributed databases. One possible use is to distribute the data in your corporate Oracle database to departmental DRDA databases. You can also distribute data in your corporate DRDA database to departmental Oracle databases.

Features

Following is a list of important features that characterize release of the Oracle gateway. The features are:

- Heterogeneous Services Architecture
- Performance Enhancements
- Fetch Reblocking
- Oracle Database 11g Passthrough Supported
- Retrieving Result Sets Through Passthrough
- Support for TCP/IP
- Native Semantics
- Columns Supported in a Result Set
- EXPLAIN_PLAN Improvement
- Heterogeneous Database IntegrationMinimum Impact on Existing Systems
- Large Base of Data Access
- Application Portability
- Remote Data Access
- Support for Distributed Applications
- Application Development and End User Tools
- Password Encryption Utility
- Support for DB2/OS390 V6, V7 and V8 Stored Procedures

- Codepage Map Facility
- IBM DB2 Universal Database Support
- IBM DB2 Version 5.1 ASCII Tables
- Read-Only Support
- Support for Graphic and Multi-byte Data
- Support for DB2/UDB on Intel Hardware
- Data Dictionary Support for DB2/UDB

Heterogeneous Services Architecture

The 11.1 release of the Oracle Database Gateway for DRDA utilizes the Oracle Heterogeneous Services component within the Oracle Database 11*g*. Heterogeneous Services is the building block for the next generation of Oracle Open Gateways.

For detailed information about heterogeneous services, refer to *Oracle Database Heterogeneous Connectivity Administrator's Guide*.

Performance Enhancements

Oracle Database Gateway for DRDA contains several internal performance enhancements. This product has shown major improvements in response time and CPU utilization for all relevant address spaces for a variety of workloads compared to version 9 gateways. The actual performance improvement at your site might vary, depending on your installation type and workload.

Fetch Reblocking

The array size of the application for `SELECT` is effective between the application and the Oracle database. However, the array blocksize and the block fetch between the Oracle database and the gateway are controlled by two Heterogeneous Services initialization parameters: `HS_RPC_FETCH_SIZE` and `HS_RPC_FETCH_REBLOCKING`. These parameters are specified in the gateway initialization file. Refer to *Oracle Database Heterogeneous Connectivity Administrator's Guide* for more information.

Oracle Database 11*g* Passthrough Supported

You can use the Oracle Database 11*g* `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` method to pass commands or statements available in your DRDA database through the gateway.

Retrieving Result Sets Through Passthrough

Oracle Database Gateway for DRDA provides a facility to retrieve result sets from a select SQL statement issued with passthrough. Refer to "Retrieving Results Sets Through Passthrough" on page 4-25 for additional information.

Support for TCP/IP

This release of the gateway supports the TCP/IP communication protocol between the gateway and the DRDA server. Refer to Chapter 14, Configuring Oracle Database Gateway for DRDA, in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64*, or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* depending on your platform.

Native Semantics

This release of the gateway supports the ability to selectively enable or disable post-processing of various SQL functions by the DRDA server. Refer to "Native Semantics" on page 4-16 for further information.

Columns Supported in a Result Set

Oracle Database Gateway for DRDA supports up to 1000 columns in a result set.

EXPLAIN_PLAN Improvement

The `EXPLAIN_PLAN` table contains the actual SQL statements passed to the DRDA server from the Oracle Database 11*g* through the gateway.

Heterogeneous Database Integration

The gateway support for ANSI-standard SQL enables read/write access to DRDA databases. Even if your data exists on different platforms in different applications, new applications can use all data, regardless of location.

Minimum Impact on Existing Systems

The gateway does not require installation of additional Oracle software on your OS/390 (MVS), AS/400, UNIX or Microsoft Windows target system. The database interface that it uses is provided by IBM and is built into the DRDA database products and network facilities that already exist on these platforms.

Configuring an IBM system for DRDA access typically consists of defining the network resources involved and establishing access security definitions specific to the target database.

Large Base of Data Access

DRDA Application Server Function is supported by most IBM DB2 database products.

Application Portability

The gateway's ability to interface with heterogeneous databases makes it possible to develop a single set of portable applications that can be used against both Oracle and IBM databases, and any other databases for which Oracle provides gateways.

Remote Data Access

Location flexibility is maximized because the gateway architecture permits network connections between each of the components. The application can use the Oracle client-server capability to connect to a remote Oracle database through Oracle Net. The Oracle database can connect to a remote gateway using a database link. The gateway connects to DRDA server through network facilities.

The benefits of remote access are:

- Provides a means to allocate the appropriate resource to a given task

  You can, for example, move application development off expensive processors and onto cost-efficient workstations or microcomputers.

- Expands the number of available data sources

  Without remote access, you are limited to the data available in the local environment. With remote access, your data sources are limited only by your networks.

- Provides a means to tailor an application environment to a given user

For example, some users prefer a block-mode terminal environment, while others prefer a bit-mapped, graphics driven terminal environment. Remote access can satisfy both because you are not constrained by the interface environment imposed by the location of your data.

Support for Distributed Applications

Because the gateway gives your application direct access to DRDA data, you eliminate the need to upload and download large quantities of database data to other processors. Instead, you can access data where it is, when you want it, without having to move the data between machines and risk unsynchronized and inconsistent data. Avoiding massive data replication can also reduce aggregate disk storage requirements over all your systems.

However, if your system design requires moving data among the machines in a network, SQL*Plus and the gateway can simplify the data transfer. With a single SQL*Plus command, you can move entire sets of data from one node of the network to another and from one database to another.

You can pass commands and statements specific to your DRDA database through the gateway to be executed by the DRDA database. For example, you can pass native DB2 SQL through the gateway for DB2 to execute. You can also execute stored procedures defined in non-Oracle databases.

Application Development and End User Tools

Through the gateway, Oracle extends the range of application development and end-user tools you can use to access your IBM databases. These tools increase application development and user productivity by reducing prototype, development, and maintenance time. Current Oracle database users do not have to learn a new set of tools to access data stored in DRDA databases. Instead, they can access Oracle database and DRDA data with a single set of tools.

With the gateway and the application development tools available from Oracle you can develop a single set of applications to access Oracle database and DRDA data. Users can use the decision support tools available from Oracle to access Oracle database and DRDA data. These tools can run on remote machines connected through Oracle Net to the Oracle database.

When designing applications, keep in mind that the gateway is designed for retrieval and relatively light transaction loads. The gateway is not currently designed to be a heavy transaction processing system.

Password Encryption Utility

The 11.1 release of the gateway includes a utility to support encryption of plain-text passwords in the Gateway Initialization File. Refer to Chapter 15, "Security Considerations" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* for details.

Support for DB2/OS390 V6, V7 and V8 Stored Procedures

The 11.1 release of the gateway supports the native stored procedure catalogs in DB2 V6, V7 and V8 (`SYSIBM.SYSROUTINES` and `SYSIBM.SYSPARMS`).

Codepage Map Facility

This release of the gateway supports external mapping of IBM CCSIDs to Oracle database character sets. Refer to "Gateway Codepage Map Facility" on page C-6.

IBM DB2 Universal Database Support

The 11.1 release supports IBM DB2 Universal Database.

IBM DB2 Version 5.1 ASCII Tables

IBM DB2 Version 5.1 supports ASCII and EBCDIC character sets. The character set selection is defined during table creation. The Oracle Database Gateway for DRDA supports access to EBCDIC tables and ASCII tables. Refer to Appendix C, "Globalization Support for DRDA".

Read-Only Support

The 11.1 release enables the gateway to be configured as a read-only gateway. In this mode, no modifying of user data will be allowed. For more information, refer to "DRDA_READ_ONLY" on page B-16.

Support for Graphic and Multi-byte Data

The 11.1 release of the gateway adds support for DB2 `GRAPHIC` and `VARGRAPHIC` data types. Refer to Chapter 4, "Developing Applications".

Support for DB2/UDB on Intel Hardware

The 11.1 release of the gateway adds support for DRDA servers running on Microsoft Windows and Linux on Intel hardware.

Data Dictionary Support for DB2/UDB

The 11.1 release of the gateway adds Oracle data dictionary support for DB2 UDB V7.

2

Release Information

This chapter provides information specific to the 11*g* release of the Oracle Database Gateway for DRDA. It includes the following sections:

Product Set

The following production components are included on the product installation media:

- Oracle Database Gateway for DRDA, release 11.1.0.5.0
- Oracle Net, release 11.1.0.5.0

Changes and Enhancements

The following are the changes and enhancements unique to the 11g release of the gateway.

SNA protocol Desupported

Protocol support for SNA network communication has been discontinued. Customers should migrate existing installations to use the TCP/IP protocol.

SQL/DS and DB2 on VM targets Desupported

Support for access to SQL/DS or DB2 on VM has been discontinued.

Gateway Password Encryption Tool

The Gateway Password Encryption tool (`g4drpwd`) has been replaced by a generic feature which is now part of Heterogeneous Services. Refer to Chapter 15, "Security Considerations" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* for details..

Product Migration

Refer to Chapter 16, "Migration From Previous Releases" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC*

(64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64 or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* for information on migrating product configurations from previous releases for additional changes or requirements.

Known Problems

The problems that are documented in the following section are specific to the Oracle Database Gateway for DRDA, and are known to exist in this release of the product. These problems will be fixed in a future gateway release. If you have any questions or concerns about these problems, contact Oracle Support Services.

A current list of problems is available online. Contact your local Oracle office for information about accessing this online information.

Known Restrictions

The following restrictions are known to exist for the products in the 11g release. Restrictions are not scheduled to change in future releases. Refer to Chapter 4, "Developing Applications", for information or limitations when developing your applications.

Accessing DB2 Alias Objects

If you need to access DB2 alias objects on a remote DB2 system, then you must specify `DRDA_DESCRIBE_TABLE=FALSE` initialization parameter in the gateway initialization file.

Oracle SQL Command INSERT

When copying data from an Oracle database to a DRDA server, the Oracle SQL command `INSERT` is not supported. The SQL*Plus `COPY` command must be used. Refer to Chapter 3, "Using the Oracle Database Gateway for DRDA", for more information.

The following are the considerations and limitations:

Stored Procedure and User Defined Function Support

The gateway supports execution of stored procedures and user defined functions through the following DRDA servers:

DB2/OS390 V4.1 or later

DB2/400 V3.1 or later

DB2/UDB V7.1 or later

DB2 Considerations

The following considerations are exist in the 11*g* release:

DD Basic Tables and Views

The owner of DD basic tables and views is `OTGDB2`. This cannot be changed.

SUBSTR Function Post-Processed

The `SUBSTR` function can be used with the Oracle database in ways that are not compatible with a DRDA server database, such as DB2/OS390. Therefore, the `SUBSTR`

function is post-processed. However, it is possible to allow the server to process it natively using the "Native Semantics" feature. Refer to Chapter 4, "Developing Applications", for details.

Support for DRDA Server Character Sets

Support for character sets used by a DRDA server is configurable through the gateways Codepage Map Facility. Refer to Appendix C, "Globalization Support for DRDA" for more information.

Data type Limitations

Refer to "DRDA Data type to Oracle Data type Conversion" on page 4-18 for detailed information about data types.

SAVEPOINT Command Is Not Supported

Oracle Database Gateway for DRDA does not support the SQL `SAVEPOINT`.

Null Values and Stored Procedures

Null values are not passed into, or returned from, calls to stored procedures through the gateway.

String Concatenation of Numbers

String concatenation of numbers is not allowed in DB2/400, DB2/UDB, and DB2/OS390. For example, `2||2` is not allowed.

GLOBAL_NAMES Initialization Parameter

If `GLOBAL_NAME`S is set to `TRUE` in the Oracle database `INIT.ORA` file, then in order to be able to connect to the gateway, you must specify the Heterogeneous Services (HS) initialization parameter, `HS_DB_DOMAIN`, in the Gateway Initialization Parameter file to match the value of the `DB_DOMAIN` parameter of the Oracle database. Refer to Chapter 14, "Configuring Oracle Database Gateway for DRDA" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* , depending on your platform, for more information.

Binding the DRDA Package on DB2/UDB

The DRDA gateway package must be bound on the DRDA server before the gateway can perform any SQL operations. Because of a DB2/UDB restriction, the `ORACLE2PC` table must be created in the DB2/UDB database before the package can be bound. Refer to Chapter 14, "Configuring Oracle Database Gateway for DRDA" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* , depending on your platform, for more information.

Date Arithmetic

In general, the following types of SQL expression forms do not work correctly with the gateway because of DRDA Serve limitations:

```
date + number
number + date
date - number
date1 - date2
```

DRDA server does not allow number addition or subtraction with date data types. The date and number addition and subtraction *(date + number, number + date, date - number)* forms are sent through to the DRDA server where they are rejected.

Also, DRDA server does not perform date subtraction consistently. When you subtract two dates *(date1 - date2)*, differing interpretations of date subtraction in the DRDA server cause the results to vary by server.

> **Note:** Avoid date arithmetic expressions in all gateway SQL until date arithmetic problems are resolved.

Row Length Limitation

Because of a restriction of the DRDA architecture, rows with aggregate length exceeding 32KB in DRDA representation cannot be stored or retrieved.

LONG Data type in SQL*Plus

SQL*Plus cannot fetch `LONG` columns from the Oracle Database Gateway for DRDA.

Single Gateway Instance per DRDA Network Interface

When installing the gateway, a proper DRDA Network Interface must be chosen. Only one DRDA Network Interface may be chosen and installed per gateway instance. If the gateway product is reinstalled, and if a Network Interface different from the previous installation is chosen, then the new choice will overlay the current installation. Reconfiguration of the Gateway Initialization Parameters must occur at this point in order to ensure proper gateway operation.

Stored Procedures and Transaction Integrity

IBM DB2 has introduced a feature called Commit on Return for stored procedures. This feature allows DB2 to perform an automatic commit after a stored procedure runs successfully. This feature is enabled when the procedure is created. To ensure data integrity, this feature is not supported by the Oracle Database Gateway for DRDA in a heterogeneous environment. When attempting to call a stored procedure which has this feature enabled, through the gateway, the gateway will return an error, `ORA-28526` or `PLS-00201` (identifier must be declared).

> **Note:** This restriction applies to DB2 for MVS or z/OS as of V5.1 and DB2/UDB as of V8.1.

SQL Limitations

The SQL limitations are described as follows:

Oracle ROWID Column

The `DB2 ROWID` column is not compatible with the Oracle `ROWID` column. Because the `ROWID` column is not supported, the following restrictions apply:

- `UPDATE` and `DELETE` are not supported with the `WHERE CURRENT OF CURSOR` clause. To update or delete a specific row through the gateway, a condition style `WHERE` clause must be used. (Bug No. 205538)

When UPDATE and DELETE statements are used in precompiler and PL/SQL programs, they rely internally on the Oracle ROWID function.

- Snapshots between Oracle database and DB2 are not supported.

  Snapshots rely internally on the Oracle ROWID column.

Oracle Bind Variables

Oracle bind variables become SQL parameter markers when used with the gateway. Therefore, the bind variables are subject to the same restrictions as SQL parameter markers.

For example, the following statements are not allowed:

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

CONNECT BY Is Not Supported

Oracle Database Gateway for DRDA does not support CONNECT BY in SELECT statements.

COUNT Function Compatibility

The following DRDA servers do not support all forms of the COUNT function, specifically COUNT(colname) and COUNT(ALL colname):

DB2 OS/390 V6,

The default for all DRDA server platforms is for all forms of COUNT to be passed to the DRDA server as it is.

If the gateway is to be used with one of the releases of DRDA servers, then it may be necessary to disable the default usage of this form of COUNT.

Refer to Chapter , "Mapping the COUNT Function" and Chapter , "Native Semantics" for details on how to disable or enable compatibility for these forms of COUNT.

3

Using the Oracle Database Gateway for DRDA

Using the Oracle Database Gateway for DRDA involves connecting to the corresponding gateway system and the remote DRDA database associated with the gateway. It is important to understand how to process and use database links. Database links are discussed in detail in the *Oracle Database Reference*. Read the database link information in that guide to understand database link processing. Then proceed to read this chapter to understand how to set up a database link to a remote DRDA database.

This chapter contains the following sections:

Processing a Database Link

The database and application administrators of a distributed database system are responsible for managing the database links that define paths to the DRDA database. The tasks are as follows:

Creating Database Links

To create a database link and define a path to a remote database, 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 remote database. The USING clause points to a tnsnames.ora connect descriptor.

> **Note:** If you do not specify a user ID and a password in the CONNECT TO clause, then the Oracle database user ID and password are used.

> **See Also:** " Refer to Chapter 15, "Security Considerations" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* for details.

The following example creates a database link to access information in the DRDA server database:

```
CREATE PUBLIC DATABASE LINK dblink
CONNECT TO userid IDENTIFIED BY password
USING 'tns_name_entry';
```

where:

dblink is the complete database link name.

user id is the user ID used to establish a session in the remote database. This user ID must be a valid DRDA server user ID. It must be authorized to any table or file on the DRDA server that is referenced in the SQL commands. The user ID must be lesser than eight characters.

password is the password used to establish a session in the remote database. This password must be a valid DRDA server password. The password must be lesser than eight characters.

tns_name_entry specifies the Oracle Net connect descriptor used to identify the gateway.

Guidelines for Database Links

Database links are active for the duration of a gateway session. If you want to close a database link during a session, then use the ALTER session statement.

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, use the statement:

```
DROP PUBLIC DATABASE LINK dblink;
```

> **Note:** A database link should not be dropped if it is required to resolve an in-doubt distributed transaction. Refer to *Oracle Database Administrator's Guide* for additional information about dropping database links.

See Also: *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 defined database links. The ALL_DB_LINKS data dictionary views show all accessible (public and private) database links.

Limiting the Number of Active Database Links

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

Accessing the Gateway

To access the gateway, complete the following steps on the Oracle database:

1. Login to the Oracle Database
2. Creating a database Link to the DRDA Database
3. Retrieve data from the DRDA Database

Login to the Oracle Database

Log in to the Oracle database to access the gateway

Creating a database Link to the DRDA Database

For example, use:

```
CREATE PUBLIC DATABASE LINK DRDA
CONNECT TO ORADRDA IDENTIFIED BY oracle_pw
USING 'tns_name_entry'
```

Retrieve data from the DRDA Database

This query fetches the TABLE file in the library SECURE, using the name ORACLE as the DRDA server user profile. The ORACLE user profile must have the appropriate privileges on the DRDA server to access the SECURE.TABLE files:

```
SELECT * FROM SECURE.TABLE@DRDA
```

The following is an example of the error messages that are displayed if insufficient privileges are displayed:

```
ORA-1031:   insufficient privileges
DG4DRDA V11.1.0.5.0 grc=0, drc=-777 (83TC,0000), errp=ARIXO,
sqlcode=-551, sqlstate=42501, errd=FFFFFF9C,0,0,0,0,0
errmc=USER SELECT SECURE.TABLE
```

Accessing AS/400 File Members

Nothing specific to DRDA or to the gateway controls the access to AS/400 files and file members. However, DB2/400 uses a naming convention that implies that the file

member name is the same as the name of the file being addressed. For example, accessing `schema.table` implies that table is the file name and also that table is the file member name being accessed.

To access file members with names that differ from the associated file name, you must create a view within the file so that DB2/400 can reference the correct file member.

One method for creating this view involves issuing the console command `Create Logical File` (`CRTLF`). This action creates a logical association between the file name and the file member name.

> **See Also:** For additional information, refer to the AS/400 Command documentation or to the DB2/400 SQL reference document.

Using the Synonym Feature

You can provide complete data, location, and network transparency by using the synonym feature of Oracle database. When a synonym is defined, the user need not know the underlying table or network protocol being used. A synonym can be public, which means it is available to all Oracle users. A synonym can also be defined as private, available only to the user who created it. Refer to *Oracle Database Reference* for details on the synonym feature.

The following statement creates a system-wide synonym for the `EMP` file in the DRDA server with ownership of `ORACLE`:

```
CREATE PUBLIC SYNONYM EMP FOR ORACLE.EMP@DRDA
```

Performing Distributed Queries

The Oracle Database Gateway technology enables the execution of distributed queries that join Oracle database and DRDA servers and any other data store for which Oracle provides a gateway. These complex operations can be completely transparent to the users requesting the data.

The distributed query optimizer (DQO) capability can provide better performance of distributed queries. Statistical data regarding tables from DRDA server is retrieved and passed to the Oracle database. The DQO capability is enabled or disabled by the `DRDA_OPTIMIZE_QUERY` parameter. Refer to "DRDA_OPTIMIZE_QUERY" on page B-14.

The following example joins data between an Oracle database, DB2/OS390, and a DRDA server:

```
SELECT o.custname, p.projno, e.ename, sum(e.rate*p.hours)
FROM orders@DB2 o, EMP@ORACLE7 e, projects@DRDA p
WHERE o.projno = p.projno
AND p.empno = e.empno
GROUP BY o.custname, p.projno, e.ename
```

A combination of views and synonyms, using the following SQL statements, keeps the process of distributed queries transparent to the user:

```
CREATE SYNONYM orders for orders@DB2;
CREATE SYNONYM PROJECTS for PROJECTS@DRDA;
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, e.ename
```

The following SQL statement retrieves information from these three data stores in one command:

```
SELECT * FROM DETAILS;
```

The results of this command are:

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

Two-Phase Commit Processing

To fully participate in a two-phase commit transaction, a server must support the `PREPARE TRANSACTION` statement. The `PREPARE TRANSACTION` statement ensures that all participating databases are prepared to `COMMIT` or to `ROLLBACK` a specific unit of work.

Oracle database supports the `PREPARE TRANSACTION` statement. Any number of Oracle database can participate in a distributed two-phase commit transaction. The `PREPARE TRANSACTION` statement is performed automatically when a `COMMIT` is issued explicitly by an application or implicitly at the normal end of the application.

The gateway does not support the `PREPARE TRANSACTION` statement. This limits the two-phase commit protocol when the gateway participates in a distributed transaction. The gateway becomes the commit focal point site of a distributed transaction. The gateway is configured as commit/confirm, so it is always the commit point site, regardless of the commit point strength setting. The gateway commits the unit of work after verifying that all Oracle databases in the transaction have successfully committed their work. The gateway must coordinate the distributed transaction, so only one gateway can participate in an two-phase commit transaction.

Two-phase commit transactions are recorded in the `ORADRDA.ORACLE2PC` table, which is created during installation. This table is created when the o2pc.sql script is run. The owner of this table also owns the package. Refer to "DRDA Gateway Package Binding Considerations" on *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows*, depending on your platform, for more information.

Distributed DRDA Transactions

Because the `ORACLE2PC` table is used to record the status of a gateway transaction, this table must be in at the database where the DRDA update takes place. Therefore, all updates that take place over the gateway must be local to the IBM database.

> **Note:** Updates to the `ORACLE2PC` table cannot be part of an IBM distributed transaction.

For additional information about the two-phase commit process, refer to *Oracle Database Administrator's Guide*.

Read-Only Gateway

The read-only option can provide improved performance and security. This improved performance depends on your configuration and parameter selections. A Gateway Initialization Parameter, DRDA_READ_ONLY, controls whether the gateway is enabled in this mode.

If you enable the read-only option, then only queries (SELECT statements) are allowed by the gateway. The capabilities that control whether updates are allowed by 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 on the DRDA server are rejected.

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

Replicating in a Heterogeneous Environment

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

Oracle Database 11*g* Triggers

When updates are made to Oracle database, synchronous copies of Oracle and non-Oracle data can be maintained automatically by using Oracle Database 11*g* triggers.

Oracle Snapshots

Oracle Database Gateway for DRDA can use the Oracle snapshot feature to automatically replicate non-Oracle data into Oracle database. The complete refresh capability of Oracle snapshot can be used to propagate a complete copy or a subset of the non-Oracle data into Oracle database at user-defined intervals.

Copying Data from Oracle Database 11*g* to DRDA Server

The COPY command enables you to copy data from Oracle database to a DRDA server. The Oracle SQL command INSERT is not supported. If you use the INSERT command:

```
INSERT INTO DRDA_table SELECT * FROM local_table
```

then the following message is displayed:

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

To copy data from your Oracle database to the DRDA server, use:

```
COPY FROM username/password@connect_identifier -
INSERT destination_table -
USING query
```

For example, to select all rows from the local Oracle EMP table, insert them into the EMP table on the DRDA server, and commit the transaction, use:

```
COPY FROM scott/tiger@ORACLE -
INSERT scott.EMP@DRDA -
USING SELECT * FROM EMP
```

The SQL*Plus COPY command supports APPEND, CREATE, INSERT, and REPLACE commands. However, INSERT is the only command supported when copying to the DRDA server. For more information about the COPY command, refer to *SQL*Plus User's Guide and Reference*.

Copying Data from DRDA Server to Oracle Database 11*g*

The CREATE TABLE command enables you to copy data from a DRDA server to Oracle database. To create a table on your Oracle database and to insert rows from a DRDA server table, use:

```
CREATE TABLE table_name
AS query
```

The following example creates the table EMP in your local Oracle database and inserts the rows from the EMP table on the DRDA server:

```
CREATE TABLE EMP
AS SELECT * FROM scott.EMP@DRDA
```

Alternatively, you can use the SQL*Plus COPY command to copy data from a DRDA server to Oracle database. For more information about the COPY command, refer to *SQL*Plus User's Guide and Reference*.

Tracing SQL Statements

SQL statements issued through the gateway can be changed before reaching the DRDA database. These changes are made to make the format acceptable to the gateway or to make Oracle SQL compatible with DRDA server SQL. Oracle database and the gateway can change the statements depending on the situation.

For various reasons, you might need to assess whether the gateway has altered the statement correctly or whether the statement could be rewritten to improve performance. SQL tracing is a feature that allows you to view the changes made to a SQL statement by the Oracle database or the gateway.

SQL tracing reduces gateway performance. Use tracing only while testing and debugging your application. Do not enable SQL tracing when the application is running in a production environment. For more information about enabling SQL tracing, refer to the section on "SQL Tracing and the Gateway" on page 5-5 in Chapter 5, "Error Messages, Diagnosis, and Reporting".

4

Developing Applications

Oracle Database Gateway for DRDA allows applications written for Oracle database to access tables in a DRDA database. This access can be virtually transparent by using synonyms or views of the DRDA tables accessed by a database link. However, fundamental SQL, data type, and semantic differences exist between the Oracle database and DRDA databases.

This chapter provides information that is specific to the 11.1 release of the Oracle Database Gateway for DRDA. This chapter contains the following sections:

Gateway Appearance to Application Programs

An application that is written to access information in a DRDA database interfaces with an Oracle database. When developing applications, keep the following information in mind:

- You must define the DRDA database to the application by using of a database link that is defined in the Oracle database. Your application should specify tables that exist on a DRDA database by using the name that is defined in the database link. For example, assume that a database link is defined so that it names the DRDA database link `DRDA`, and also assume that an application needs to retrieve data from an Oracle database and from the `DRDA` database. Use the following SQL statement (joining two tables together) in your application:

```
SELECT EMPNO, SALARY
FROM EMP L, EMPS@DRDA R
WHERE L.EMPNO = R.EMPNO
```

In this example, EMP is a table on an Oracle database, and EMPS is a table on a DRDA server. You can also define a synonym or a view on the DRDA server table, and access the information without the database link suffix.

- You can read and write data to a defined DRDA database. SELECT, INSERT, UPDATE, and DELETE are all valid operations.
- A single transaction can write to one DRDA database and to multiple Oracle databases.
- Single SQL statements, using JOINs, can refer to tables in multiple Oracle databases, in multiple DRDA databases, or in both.

Fetch Reblocking

Oracle database 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 11*g*. The buffer may contain one or more qualified rows from the DRDA server. This feature can provide significant performance enhancements, depending on your application design, installation type, and workload.

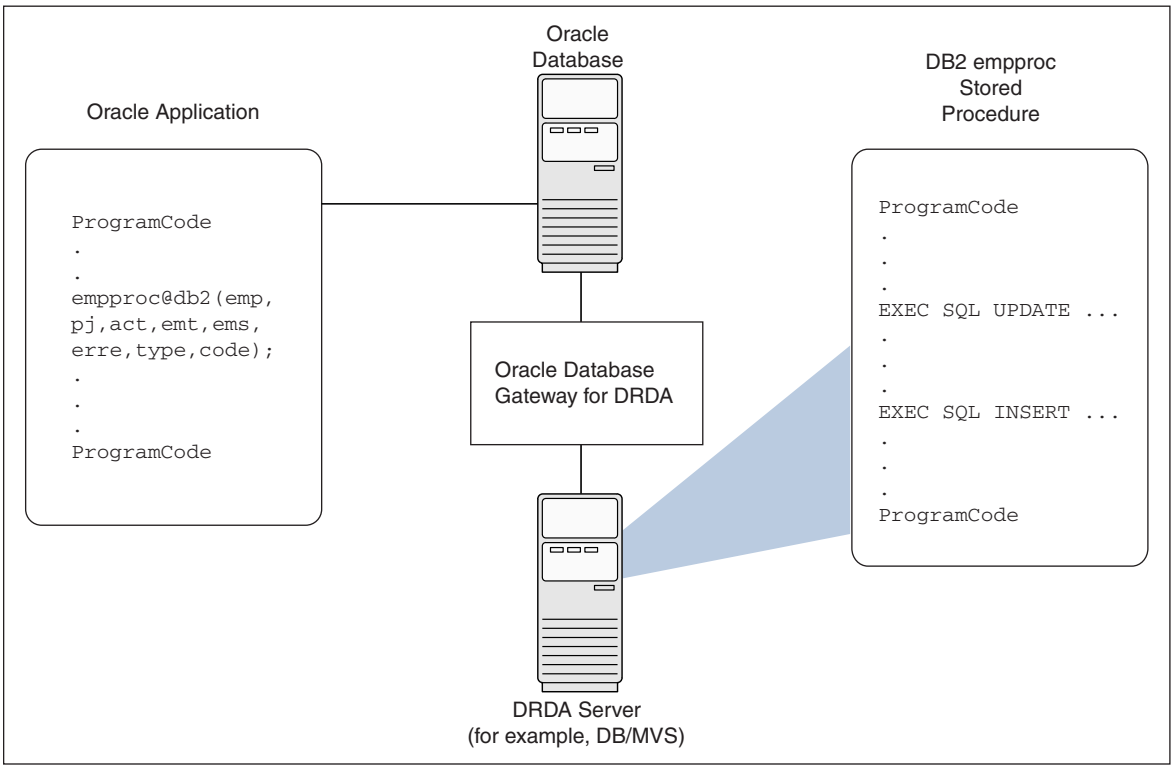
The array size between the client and the Oracle Database 11*g* is determined by the Oracle application. Refer to Chapter 14, "Configuring Oracle Database Gateway for DRDA" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* , depending on your platform, for more information.

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 together a set of SQL and other PL/SQL programming language statements 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 Oracle database and in a remote instance. Figure 4–1 illustrates two stored procedures: oraproc1 is a procedure stored in the ORA1 Oracle instance, and oraproc2 is a procedure stored in the ORA2 Oracle instance.

Figure 4–1 Calling Oracle Stored Procedures in a Distributed Oracle Environment



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 in the remote Oracle instance.

In Figure 4–1, 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.

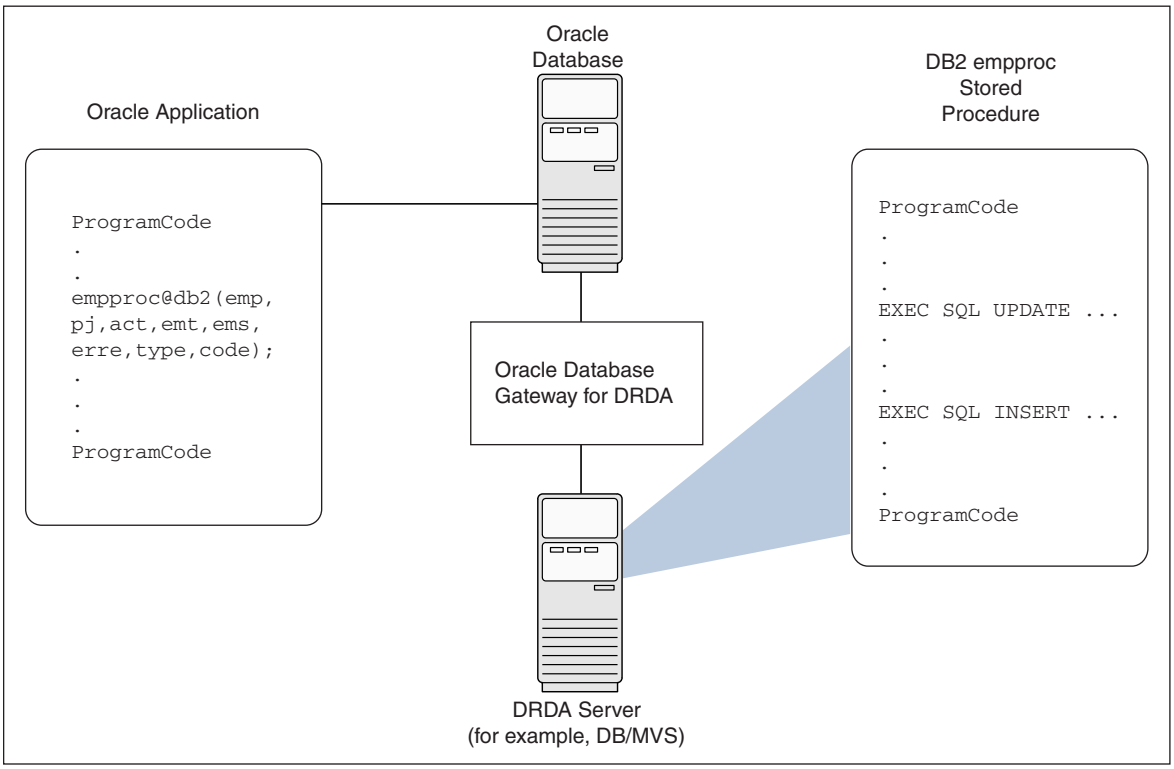
Using DRDA Server Stored Procedures with the Gateway

The procedural feature of the gateway enables invocation of native DRDA server stored procedures. In other words, the stored procedure is no longer defined in the Oracle database, but, is defined in the DRDA server (for example., DB2/OS390). Again, standard Oracle PL/SQL is used by the Oracle application to run the stored procedure.

After the stored procedure is defined to the DRDA server (for example., DB2/OS390), the gateway is able to use the existing DRDA server definition to run the procedure. The gateway does not require special definitions to call the DB2 stored procedure.

In Figure 4–2, an Oracle application calls the `empproc` stored procedure that is defined to the DRDA server (for example., DB2/OS390).

Figure 4–2 Running DRDA Server Stored Procedures



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

Oracle Application and DRDA Server Stored Procedure Completion

If an Oracle Application attempts to invoke a stored procedure in a DB2/OS390 database. For an Oracle application to call a DB2 stored procedure, it is first necessary to create the DB2 stored procedure on the DB2 system by using the procedures documented in the IBM reference document for DB2 for OS/390 SQL.

After the stored procedure is defined in 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 in order to calculate a new yearly salary for John Smythe. The revised salary that is returned as result is used to update the `EMP` table of Oracle database:

```
DECLARE
  INPUT VARCHAR2(15);
  RESULT NUMBER(8,2);
BEGIN
  INPUT := 'JOHN SMYTHE';
  REVISE_SALARY@DB2(INPUT, RESULT);
  UPDATE EMP SET SAL = RESULT WHERE ENAME = INPUT;
END;
```

When the gateway receives a call to run a stored procedure on the DRDA server (for example DB2/OS390), it first does a lookup of the procedure name in the server catalog. The information that defines a stored procedure is stored in different forms on each DRDA server. For example, DB2/OS390 V5.0 uses the table

SYSIBM.SYSPROCEDURES, while DB2/OS390 V6.1 uses the table SYSIBM.SYSROUTINES and SYSIBM.SYSPARMS, and DB2/400 uses the table QSYS2.SYSPROCS and QSYS2.SYSPARMS. The gateway has a list of known catalogs to search, depending on the DRDA server that is being accessed.

The search order of the catalogs is dependent on whether the catalogs support Location designators (such as LUNAME in SYSIBM.SYSPROCEDURES), and authorization or owner IDs (such as AUTHID in SYSIBM.SYSPROCEDURES or OWNER in SYSIBM.SYSROUTINES).

Some DRDA servers allow blank or public authorization qualifiers. If the DRDA server that is currently connected supports which form of qualification, then the gateway will apply those naming rules when searching for a procedure name in the catalog.

The matching rules will first search for a public definition, and then an owner qualified procedure name. For more detailed information, refer to the IBM reference document for DB2 for OS/390 SQL.

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 Customer Information Control System (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 if this procedure calls a CICS or IMS transaction, then the gateway cannot recover data from any activity that occurred within the CICS or IMS transaction.

  For example, the CICS transaction could roll back 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 video surveillance and monitoring (VSAM) file, then the gateway would consider this activity separate from its own recoverable unit of work.
- PL/SQL records cannot be passed as parameters when invoking a DB2 stored procedure.
- The gateway supports the SIMPLE linkage convention of DB2 stored procedures.The SIMPLE linkage convention means that the parameters that are passed to and from the DB2 stored procedure cannot be null.

Database Link Behavior

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

Oracle Database SQL Construct Processing

One of the most important features of the Oracle Open Gateways products is providing SQL transparency to the user and to the application programmer. Foreign SQL constructs can be categorized into four areas:

- Compatible
- Translated
- Compensated
- Native semantics

Compatible SQL Functions

Oracle database automatically forwards compatible SQL functions to the DRDA database. Where SQL constructs with the same syntax and meaning are on both Oracle database and the DRDA database. These SQL constructs are forwarded unmodified. All of the compatible functions are column functions. Functions that are not compatible are either translated to an equivalent DRDA SQL function or are compensated (post-processed) by Oracle database after the data is returned from the DRDA database.

Translated SQL Functions

Translated functions have the same meaning but different names between the Oracle database and the DRDA database. But all applications must use the Oracle function name. These SQL constructs that are supported with different syntax (different function names) by the DRDA database, are automatically translated by the Oracle database and then forwarded to the DRDA database. Oracle database changes the function name before sending it to the DRDA database, in a manner that is transparent to your application.

Compensated SQL Functions

Some advanced SQL constructs that are supported by Oracle database may not be supported in the same manner, if at all, by the DRDA database. Compensated functions are those SQL functions that are not recognized by the DRDA server. If a `SELECT` statement containing one of these functions is passed from the Oracle database to the gateway, then the gateway removes the function before passing the SQL statement to the DRDA server. The gateway passes the selected DRDA database rows to Oracle database. Oracle database applies the function.

Post-Processing

Oracle database can compensate for a missing or incompatible function by automatically excluding the incompatible SQL construct from the SQL request that is forwarded to the DRDA database. Oracle database then retrieves the necessary data from the DRDA database and applies the function. This process is known as post-processing.

The gateway attempts to pass all SQL functions to DRDA databases. However when a DRDA database does not support a function that is represented in the computation, the gateway changes that function. For example, if a program runs the following query against a DB2/OS390 database:

```
SELECT COS(X_COOR) FROM TABLE_X;
```

Because the database does not support many of the COS functions, the gateway changes the query to the following:

```
SELECT X_COOR FROM TABLE_X;
```

All data in the X_COOR column of TABLE_X is passed from the DB2/OS390 database to the Oracle database. After the data is moved to the Oracle database, the COS function is performed.

If you are performing operations on large amounts of data that are stored in a DRDA database, then keep in mind that some functions require post-processing.

Native Semantic SQL Functions

Some SQL functions that are normally compensated may also be overridden, through the Native Semantics facility. If a SQL function has been enabled for Native Semantics, then the function may be passed on to the DRDA database for processing, instead of being compensated (post-processed). If a SQL function is enabled for Native Semantics and is therefore passed on the DRDA database for processing, then the SQL function is processed natively in the DRDA database. Refer to "Native Semantics" on page 4-16 for more information.

DB2/OS390 SQL Compatibility

Table 13-1 describes how Oracle database and the gateway handles SQL functions for a DB2/OS390

Table 4–1 SQL Compatibility, by Oracle SQL function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
ABS	-	-	Yes	Yes
ACOS	-	-	Yes	Yes
ADD_MONTHS	-	-	Yes	
ASCII	-	-	Yes	Yes
ASIN	-	-	Yes	Yes
ATAN	-	-	Yes	Yes
ATAN2	-	-	Yes	Yes
AVG	Yes	-	-	-
BITAND	-	-	Yes	Yes
CAST	-	-	Yes	Yes
CEIL	-	CEILING	-	Yes
CHARTOROWID	-	-	Yes	-
CHR	-	-	Yes	Yes
CONCAT	Yes	-	-	-
CONVERT	-	-	Yes	Yes
COS	-	-	Yes	Yes
COSH	-	-	Yes	Yes

Table 4–1 (Cont.) SQL Compatibility, by Oracle SQL function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
COUNT(*)	Yes	-	-	-
COUNT (DISTINCT colname)	Yes	-	-	-
COUNT (ALL colname)	Yes	-	-	COUNTCOL
COUNT (column)	Yes	-	-	COUNTCOL
DECODE	-	-	Yes	Yes
DUMP	-	-	Yes	Yes
EXP	-	-	Yes	Yes
FLOOR	Yes	-	-	Yes
GREATEST	-	-	Yes	Yes
HEXTORAW	-	-	Yes	Yes
INITCAP	-	-	Yes	Yes
INSTR	-	-	Yes	Yes
INSTRB	-	-	Yes	Yes
LAST_DAY	-	-	Yes	-
LEAST	-	-	Yes	Yes
LENGTH	-	-	Yes	Yes
LENGTHB	-	-	Yes	Yes
LN	-	-	Yes	Yes
LOG	-	-	Yes	Yes
LOWER	-	-	Yes	Yes
LPAD	-	-	Yes	Yes
LTRIM	-	-	Yes	Yes
MAX	Yes	-	-	-
MIN	Yes	-	-	-
MOD	-	-	Yes	Yes
MONTHS_BETWEEN	-	-	Yes	-
NEW_TIME	-	-	Yes	-
NEXT_DAY	-	-	Yes	-
NLS_INITCAP	-	-	Yes	Yes
NLS_LOWER	-	-	Yes	Yes
NLS_UPPER		-	Yes	Yes
NLSSORT	-	-	Yes	Yes
NVL		VALUE		

Table 4–1 (Cont.) SQL Compatibility, by Oracle SQL function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
NVL2	-	-	Yes	Yes
POWER	-	-	Yes	Yes
RAWTOHEX	-	-	Yes	Yes
REPLACE	-	-	Yes	Yes
REVERSE	-	-	Yes	Yes
ROUND	-	-	Yes	Yes
ROWIDTOCHAR	-	-	Yes	-
RPAD	-	-	Yes	Yes
RTRIM	-	-	Yes	Yes
SIGN	-	-	Yes	Yes
SIN	-	-	Yes	Yes
SINH	-	-	Yes	Yes
SOUNDEX		-	Yes	-
SQRT	-	-	Yes	Yes
STDDEV	-	-	Yes	Yes
SUBSTR	-	-	Yes	Yes
SUBSTRB	-	-	Yes	Yes
SUM	Yes	-	-	-
SYSDATE	-	-	Yes	
TAN	-	-	Yes	Yes
TANH	-	-	Yes	Yes
TO_CHAR	-	-	Yes	-
TO_DATE	-	-	Yes	-
TO_MULTI_BYTE	-	-	Yes	-
TO_NUMBER	-	DECIMAL	-	Yes
TO_SINGLE_BYTE	-	-	Yes	-
TRANSLATE	-	-	Yes	Yes
TRIM	-	STRIP	Yes	Yes
TRUNC	-	-	Yes	Yes
UID	-	-	Yes	-
UPPER	-	-	Yes	Yes
USER	-	-	Yes	-
USERENV	-	-	Yes	-
VARIANCE	-	-	Yes	Yes

Table 4–1 (Cont.) SQL Compatibility, by Oracle SQL function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
VSIZE	-	-	Yes	Yes

DB2/Universal Database SQL Compatibility

The ways that Oracle database and gateway handle SQL functions for a DB2/UDB database are shown in the following table:

Table 4–2 DB2/Universal Database SQL Compatibility, by Oracle SQL Function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
ABS	Yes	-	-	Yes
ACOS	-	-	Yes	Yes
ADD_MONTHS	-	-	Yes	-
ASCII	-	-	Yes	Yes
ASIN	-	-	Yes	Yes
ATAN	-	-	Yes	Yes
ATAN2	-	-	Yes	Yes
AVG	Yes	-	-	-
BITAND	-	-	Yes	Yes
CAST	-	-	Yes	Yes
CEIL	-	CEILING	-	Yes
CHARTOROWID	-	-	Yes	-
CHR	Yes	-	-	Yes
CONCAT	Yes	-	-	-
CONVERT	-	-	Yes	Yes
COS	Yes	-	-	Yes
COSH	-	-	Yes	Yes
COUNT(*)	Yes	-	-	-
COUNT (DISTINCT colname)	Yes	-	-	-
COUNT (ALL colname)	Yes	-	-	COUNTCOL
COUNT (column)	Yes	-	-	COUNTCOL
DECODE	-	-	Yes	Yes
DUMP	-	-	Yes	Yes
EXP	Yes	-	-	Yes
FLOOR	Yes	-	-	Yes

Table 4–2 (Cont.) DB2/Universal Database SQL Compatibility, by Oracle SQL Function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
GREATEST	-	-	Yes	Yes
HEXTORAW	-	-	Yes	Yes
INITCAP	-	-	Yes	Yes
INSTR	-	-	Yes	Yes
INSTRB	-	-	Yes	Yes
LAST_DAY	-	-	Yes	-
LEAST	-	-	Yes	Yes
LENGTH	-	-	Yes	Yes
LENGTHB	-	-	Yes	Yes
LN	Yes	-	v	Yes
LOG	-	-	Yes	Yes
LOWER	-	LCASE	-	Yes
LPAD		-	Yes	Yes
LTRIM	-	-	Yes	Yes
MAX	Yes	-	-	-
MIN	Yes	-	-	-
MOD	Yes	-	-	Yes
MONTHS_BETWEEN	-	-	Yes	-
NEW_TIME	-	-	Yes	-
NEXT_DAY	Yes	-	Yes	-
NLS_INITCAP	-	-	Yes	Yes
NLS_LOWER	-	-	Yes	Yes
NLS_UPPER			Yes	Yes
NLSSORT	-	-	Yes	Yes
NVL	-	VALUE	-	-
NVL2	-	-	Yes	Yes
POWER	Yes	-	-	Yes
RAWTOHEX	-	-	Yes	Yes
REPLACE	Yes	-	-	Yes
REVERSE	-	-	Yes	Yes
ROUND	Yes	-	-	Yes
ROWIDTOCHAR		-	Yes	-
RPAD			Yes	Yes
RTRIM	-	-	Yes	Yes

Table 4–2 (Cont.) DB2/Universal Database SQL Compatibility, by Oracle SQL Function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
SIGN	Yes	-	-	Yes
SIN	Yes	-	-	Yes
SINH	-	-	Yes	Yes
SOUNDEX	-	-	Yes	-
SQRT	Yes	-	-	Yes
STDDEV	-	-	Yes	Yes
SUBSTR	-	-	Yes	Yes
SUBSTRB	-	-	Yes	Yes
SUM	Yes	-	-	-
SYSDATE	-	-	Yes	-
TAN	Yes	-	-	Yes
TANH	-	-	Yes	Yes
TO_CHAR	-	-	Yes	-
TO_DATE	-	-	Yes	-
TO_MULTI_BYTE	-	-	Yes	-
TO_NUMBER	-	DECIMAL	-	Yes
TO_SINGLE_BYTE	-	-	Yes	-
TRANSLATE	-	-	Yes	Yes
TRIM	-	-	Yes	Yes
TRUNC	Yes	-	-	Yes
UID	-	-	Yes	-
UPPER	-	UCASE	v	Yes
USER	-	-	Yes	-
USERENV	-	-	Yes	-
VARIANCE	-	-	Yes	Yes
VSIZE	-	-	Yes	Yes

DB2/400 SQL Compatibility

The ways that Oracle database and gateway handle SQL functions for a DB2/400 database are shown in the following table:

Table 4–3 DB2/400 SQL Compatibility, by Oracle SQL Function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
ABS	-	ABSVAL	-	Yes

Table 4–3 (Cont.) DB2/400 SQL Compatibility, by Oracle SQL Function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
ACOS	-	-	Yes	Yes
ADD_MONTHS	-	-	Yes	-
ASCII	-	-	Yes	Yes
ASIN	-	-	Yes	Yes
ATAN	-	-	Yes	Yes
ATAN2	-	-	Yes	Yes
AVG	Yes	-	-	-
BITAND	-	-	Yes	Yes
CAST	-	-	Yes	Yes
CEIL	-	CEILING	-	Yes
CHARTOROWID	-	-	Yes	-
CHR	-	-	Yes	Yes
CONCAT	Yes	-	-	-
CONVERT	-	-	Yes	Yes
COS	Yes	-	-	Yes
COSH	Yes	-	-	Yes
COUNT(*)	Yes	-	-	-
COUNT (DISTINCT colname)	Yes	-	-	-
COUNT (ALL colname)	Yes	-	-	COUNTCOL
COUNT (column)	Yes	-	-	COUNTCOL
DECODE	-	-	Yes	Yes
DUMP	-	-	Yes	Yes
EXP	Yes	-	-	Yes
FLOOR	Yes	-	-	Yes
GREATEST	-	-	Yes	Yes
HEXTORAW	-	-	Yes	Yes
INITCAP	-	-	Yes	Yes
INSTR	-	-	Yes	Yes
INSTRB	-	-	Yes	Yes
LAST_DAY	-	-	Yes	-
LEAST	-	-	Yes	Yes
LENGTH	-	-	Yes	Yes
LENGTHB	-	-	Yes	Yes

Table 4–3 (Cont.) DB2/400 SQL Compatibility, by Oracle SQL Function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
LN	Yes	-	-	Yes
LOG	-	-	Yes	Yes
LOWER	-	-	Yes	Yes
LPAD	-	-	Yes	Yes
LTRIM	-	-	Yes	Yes
MAX	Yes	-	-	-
MIN	Yes	-	v	-
MOD	-	-	Yes	Yes
MONTHS_BETWEEN	-	-	Yes	
NEW_TIME	-	-	Yes	
NEXT_DAX	-	-	Yes	
NLS_INITCAP	-	-	Yes	Yes
NLS_LOWER	-	-	Yes	Yes
NLS_UPPER	-	-	Yes	Yes
NLSSORT	-	-	Yes	Yes
NVL	-	VALUE	-	-
NVL2	-	-	Yes	Yes
POWER	-	--	Yes	Yes
RAWTOHEX	-	-	Yes	Yes
REPLACE	-	-	Yes	Yes
REVERSE	-	-	Yes	Yes
ROUND	-	-	Yes	Yes
ROWIDTOCHAR	-	-	Yes	-
RPAD	-	-	Yes	Yes
RTRIM	-	-	Yes	Yes
SIGN	-	-	Yes	Yes
SIN	Yes	-	-	Yes
SINH	Yes	-	-	Yes
SOUNDEX	-	-	Yes	-
SQRT	Yes	-	-	Yes
STDDEV	Yes	v	-	Yes
SUBSTR	-	-	Yes	Yes
SUBSTRB	-	-	Yes	Yes
SUM	Yes	-	-	-

Table 4–3 (Cont.) DB2/400 SQL Compatibility, by Oracle SQL Function

Oracle SQL Function	Compatible	Translated	Compensated	Native Semantics Candidate
SYSDATE	-	-	Yes	-
TAN	Yes	-	-	Yes
TANH	Yes	-	-	Yes
TO_CHAR	-	-	Yes	-
TO_DATE	-	-	Yes	-
TO_MULTI_BYTE	-	-	Yes	-
TO_NUMBER	-	DECIMAL	-	Yes
TO_SINGLE_BYTE	-	-	Yes	-
TRANSLATE	-	-	Yes	Yes
TRIM	-	-	Yes	Yes
TRUNC	-	-	Yes	Yes
UID	-	-	Yes	-
UPPER	-	TRANSLATE	-	Yes
USER	-	-	Yes	-
USERENV	-	-	Yes	-
VARIANCE	-	VAR	-	Yes
VSIZE	-	-	Yes	Yes

Native Semantics

Because some of the advanced SQL constructs that are supported by Oracle database may not be supported in the same manner by the DRDA database, the Oracle database compensates for the missing or incompatible functionality by post-processing the DRDA database data with Oracle database functionality

> **See Also:** "Oracle Database SQL Construct Processing" on page 4-6 for more information

This feature provides maximum transparency, but may impact performance. In addition, new versions of a particular DRDA database may implement previously unsupported functions or capabilities, or they may change the supported semantics as to make them more compatible with Oracle database functions.

Some of DRDA servers also provide support for user-defined functions. The user may choose to implement Oracle database functions natively (in the DRDA database). This enables the DRDA server to pass the function to the underlying database implementation (for example DB2). Native Semantics provides a method of enabling specific capabilities to be processed natively by the DRDA server.

Various considerations must be taken into account when enabling the Native Semantic feature of a particular function because Native Semantics has advantages and disadvantages, which are typically a trade-off between transparency and performance. One such consideration is the transparency of data coercions. Oracle database

provides coercion (implicit data conversion) for many SQL functions. This means that if the supplied value for a particular function is not correct, then Oracle database will coerce the value (change it to the correct value type) before processing it. However, with the Native Semantic feature enabled, the value (exactly as provided) will be passed to the DRDA server for processing. In many cases, the DRDA server will not be able to coerce the value to the correct type and will generate an error.

Another consideration involves the compatibility of parameters to a particular SQL function. For instance, Oracle database implementation of SUBSTR allows negative values for the string index, whereas most DRDA server implementations of SUBSTR do not allow negative values for the string index. However, if the application is implemented to invoke SUBSTR in a manner that is compatible with the DRDA server, then the function will behave the same in either Oracle database or the DRDA server.

Another consideration is that the processing of a function at the DRDA server may not be desirable due to resource constraints in that environment.

Refer to the "DRDA_CAPABILITY" on page B-9 for details on enabling or disabling these capabilities. Refer to the *Oracle Database SQL Language Reference* for Oracle database format of the following capabilities.

SQL Functions That Can Be Enabled

The following list contains SQL functions that are disabled (OFF) by default. They can be enabled (turned ON) as an option:

Table 4–4 List of SQL Functions That Can Be Enabled

Function Name	Function Name	Function Name	Function Name
ABS	ACOS	ASCII	ASIN
ATAN	ATAN2	BITAND	CAST
CEIL	CHR	CONVERT	COS
COSH	COUNTCOL	DECODE	DUMP
EXP	FLOOR	GREATEST	HEXOTRAW
INITCAP	INSTR	INSTRB	LEAST
LENGTH	LENGTHB	LN	LOG
LOWER	LPAD	LTRIM	MOD
NLS_INITCAP	NLS_UPPER	NLS_LOWER	NLSSORT
NVL2	POWER	RAWTOHEX	REPLACE
REVERSE	ROUND	RPAD	RTRIM
SIGN	SIN	SINH	SQRT
STDDEV	SUBSTR	SUBSTRB	TAN
TANH	TO_NUMBER	TRANSLATE	TRIM
TRUNC	UPPER	VARIANCE	VSIZE

SQL Functions That Can Be Disabled

The following SQL functions are enabled (ON) by default:

- GROUPBY
- HAVING

- ORDERBY
- WHERE

ORDERBY controls sort order, which may differ at various sort locations. For example, with ORDERBY ON, a DB2 sort would be based on Extended Binary Coded Decimal Interchange Code (EBCDIC) sorting order, whereas with ORDERBY OFF, an Oracle database sort would be based on ASCII sorting order.

The other three functions, GROUPBY, HAVING, and WHERE, can take additional processing time. If you need to minimize the use of expensive resources, then you should choose the settings of these functions so that the processing is performed with cheaper resource. The above listed functions can also be disabled.

SQL Set Operators and Clauses

The WHERE and HAVING clauses are compatible for all versions of the DRDA server. This means that these clauses are passed unchanged to the DRDA server for processing. Whether clauses GROUP BY and ORDER BY are passed to the DRDA server, or compensated by Oracle database, is determined by the Native Semantics Parameters (see the previous section).

The set operators UNION and UNION ALL are compatible for all versions of the DRDA server, meaning that they are passed unchanged to the DRDA server for processing. The set operators INTERSECT and MINUS are compensated on all versions of the DRDA server except DB2/UDB. For DB2/UDB, INTERSECT is compatible and MINUS is translated to EXCEPT.

DRDA Data type to Oracle Data type Conversion

To move data between applications and the database, the gateway binds data values from a host variable or literal of a specific data type to a data type understood by the database. Therefore, the gateway maps values from any version of the DRDA server into appropriate Oracle data types before passing these values back to the application or Oracle tool.

The following table lists the data type mapping and restrictions. The DRDA server data types that are listed in the table are general. Refer to documentation for your DRDA database for restrictions on data type size and value limitations.

Table 4–5 data type Mapping and Restrictions

DRDA server	Oracle External	Criteria
CHAR(N)	CHAR(N)	N 255
VARCHAR (N)	VARCHAR2(N) LONG	N 2000 2000 < N 32740
LONG VARCHAR(N)	VARCHAR2(N)	N 2000
LONG VARCHAR(N)	LONG	2000 < N <= 32740
CHAR(N) FOR BIT DATA	RAW(N)	N 255
VARCHAR(N) FOR BIT DATA	RAW(N)	1 N 255
VARCHAR(N) FOR BIT DATA	LONG RAW(N)	255 < N 32740
LONG VARCHAR(N) FOR BIT DATA	RAW(N)	1 N 255

Table 4–5 (Cont.) data type Mapping and Restrictions

DRDA server	Oracle External	Criteria
LONG VARCHAR(N) FOR BIT DATA	LONG RAW(N)	255 < N 32740
DATE	DATE	Refer to Performing Date and Time Operations
TIME	CHAR(8)	See Performing Date and Time Operations
TIMESTAMP	CHAR(26)	See Performing Date and Time Operations
GRAPHIC	CHAR(2N)	N <= 127
VARGRAPHIC	VARCHAR2(2N) LONG	N <= 1000 1000 <= N <= 16370
LONG VARGRAPHIC	VARCHAR2(2N) LONG	N <= 1000 1000 <= N <= 16370
Floating Point Single	FLOAT(21)	n/a
Floating Point Double	FLOAT(53)	n/a
Decimal (P, S)	NUMBER(P,S)	n/a

Performing Character String Operations

The gateway performs all character string comparisons, concatenations, and sorts using the data type of the referenced columns, and determines the validity of character string values passed by applications using the gateway. The gateway automatically converts character strings from one data type to another and converts between character strings and dates when needed.

Frequently, DRDA databases are designed to hold non-character binary data in character columns. Applications executed on DRDA systems can generally store and retrieve data as though it contained character data. However, when an application accessing this data runs in an environment that uses a different character set, inaccurate data may be returned.

With the gateway running on the host, character data retrieved from a DB2/400 or DB2/OS390host is translated from EBCDIC to ASCII. When character data is sent to DB2/400 or DB2/OS390 from the host, ASCII data is translated to EBCDIC. When the characters are binary data in a character column, this translation causes the application to receive incorrect information or errors. To resolve these errors, character columns on DB2/400 or DB2/OS390 that hold non-character data must be created with the FOR BIT DATA option. In the application, the character columns holding non-character data should 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.

Converting Character String Data types

The gateway binds character string data values from host variables as fixed-length character strings. The bind length is the length of the character string data value. The gateway performs this conversion on every bind.

The DRDA VARCHAR data type can be from 1 to 32740 bytes in length. This data type is converted to an Oracle VARCHAR2 data type if it is between 1 and 2000 characters in

length. If it is between 2000 and 32740 characters in length, then it is converted to an Oracle LONG data type.

The DRDA VARCHAR data type can be no longer than 32740 bytes, which is much shorter than the maximum size for the Oracle LONG data type. If you define an Oracle LONG data type larger than 32740 bytes in length, then you receive an error message when it is mapped to the DRDA VARCHAR data type.

Performing Graphic String Operations

DB2 GRAPHIC data types store only double-byte string data. Sizes for DB2 GRAPHIC data types typically have maximum sizes that are half that of their Character counterparts. For example, the maximum size of a CHAR may be 255 characters, whereas the maximum size of a GRAPHIC may be 127 characters.

Oracle database does not have a direct matching data type, and the gateway therefore converts between Oracle character data types to DB2 Graphic data types. Oracle database character data types may contain single, mixed, or double-byte character data. The gateway converts the string data into appropriate double-byte-only format depending upon whether the target DB2 column is a Graphic type and whether Gateway Initialization parameters are set to perform this conversion. For more configuration information, refer to Appendix B, "Initialization Parameters" and Appendix C, "Globalization Support for DRDA".

Performing Date and Time Operations

The implementation of date and time data differs significantly in IBM DRDA databases and Oracle database. Oracle database has a single date data type, DATE, that can contain both calendar date and time of day information.

IBM DRDA databases support the following three distinct date and time data types:

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 in the internal format of the IBM DRDA database.

Processing TIME and TIMESTAMP Data

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 to 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 functions as character strings and converts or subsets portions of the string to perform numerical operations. TIME and TIMESTAMP values can be sent to an IBM DRDA database as character literals or bind variables of the appropriate length and format.

Processing DATE Data

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 Oracle format and IBM DRDA 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 an IBM DRDA compatible format.

The gateway does not automatically recognize when a character value is being processed against an IBM DATE column. Applications are required to distinguish character date values by enclosing them with Oracle TO_DATE function notation. For example, if EMP is a synonym or view that accesses data on an IBM DRDA database, then you should not use the following SQL statement:

```
SELECT * FROM EMP WHERE HIREDATE = '03-MAR-81'
```

you should use the following:

```
SELECT * FROM EMP WHERE HIREDATE = TO_DATE('03-MAR-81')
```

In a programmatic interface program that uses a character bind variable for the qualifying date value, you must use this SQL statement:

```
SELECT * FROM EMP WHERE HIREDATE = TO_DATE(:1)
```

The above SQL notation does not affect SQL statement semantics when the statement is executed against an Oracle database table. The statement remains portable across Oracle and IBM DRDA-accessed data stores.

The TO_DATE function is not required for dates in any of the following formats:

- YYYY-MM-DD (ISO/JIS)
- DD.MM.YYYY (European)
- MM/DD/YYYY (USA)

For example:

```
SELECT * FROM EMP WHERE HIREDATE = '1981-03-03'
```

The TO_DATE requirement also does not pertain to input bind variables that are in Oracle date 7-byte binary format. The gateway recognizes such values as dates.

Performing Date Arithmetic

The following forms of SQL expression generally 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 DRDA server, where they are rejected. The supported servers do not permit number addition or subtraction with dates.

Because of differing interpretations of date subtraction in the supported servers, subtracting two dates *(date1 - date2)* gives results that vary by server.

> **Note:** Avoid date arithmetic expressions in all gateway SQL until date arithmetic problems are resolved.

Dates

Date handling has two categories:

- Two-digit year dates, which are treated as occurring 50 years before or 50 years after the year 2000.
- Four-digit year dates, which are not ambiguous with regard to the year 2000.

Oracle recommends that you set the Oracle Database 11*g* server and gateway default HS_NLS_DATE_FORMAT parameter to a format including a four-digit year.

Use one of the following methods to enter twenty-first century dates:

- The TO_DATE function

  Use any date format including a four character year field. Refer to the *Oracle Database SQL Language Reference* for the available date format string options.

  For example, TO_DATE('2008-07-23', 'YYYY-MM-DD') can be used in any SELECT, INSERT, UPDATE, or DELETE statement.

- The HS_NLS_DATE_FORMAT parameter

  The HS_NLS_DATE_FORMAT parameter defines a default format for Oracle database explicit TO_DATE functions without a pattern and for implicit string to date conversions.

  For example, with HS_NLS_DATE_FORMAT defined as 'YYYY-MM-DD', '2008-07-23' can be used in any SELECT, INSERT, UPDATE, or DELETE statement.

HS_NLS_DATE_FORMAT Support

The following table lists the four patterns that can be used for the HS_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 database default format of 'DD-MON-YY' is not permitted with DB2.

The following example demonstrates how to enter and select date values in the twenty-first century:

```
ALTER SESSION SET HS_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. Only TO_DATE(literal) or TO_DATE(:bind_variable) is permitted. Except in SELECT result lists, the TO_DATE(column_name) function format is not supported.

The preprocessing of the Oracle TO_DATE functions into simple values is useful in an INSERT VALUES clause because DB2 does not allow functions in the VALUES clause. In this case, DB2 receives a simple value in the VALUES list. All forms of the TO_DATE function (with one, two, or three operands) are supported.

Performing Numeric data type Operations

IBM versions of the DRDA server perform automatic conversions to the numeric data type of the destination column (such as integer, double-precision floating point, or decimal). The user has no control over the data type conversion, and this conversion can 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 an IBM DRDA database, then the update shown in the following example inaccurately sets the price of an ice cream cone to $1.00 because the IBM DRDA 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 IBM DRDA server automatically converts the decimal data value of 1.50 to 1.

Mapping the COUNT Function

Oracle database supports the following four operands for the COUNT function:

- COUNT(*)
- COUNT(DISTINCT colname)
- COUNT(ALL colname)
- COUNT(colname)

Some DRDA servers do not support all forms of COUNT, specifically COUNT(colname) and COUNT(ALL colname). In those cases the COUNT function and its arguments are translated into COUNT(*). This may not yield the desired results, especially if the column being counted contains NULLs.

For those DRDA servers that do not support the above forms, it may be possible to achieve equivalent functionality by adding a WHERE clause. For example,

```
SELECT COUNT(colname) FROM table@dblink WHERE colname IS NOT NULL
```

or

```
SELECT COUNT(ALL colname) FROM table@dblink WHERE colname IS NOT NULL
```

Refer to Chapter , "SQL Limitations" for known DRDA servers which do not support all forms of COUNT.

Performing Zoned Decimal Operations

A zoned decimal field is described as packed decimal on Oracle database. However, an Oracle application such as a Pro*C program can insert into a zoned decimal column using any supported Oracle numeric data type. The gateway converts this number

into the most suitable data type. Data can be fetched from a DRDA database into any Oracle data type, provided that it does not result in a loss of information.

Passing Native SQL Statements through the Gateway

The passthrough SQL feature enables an application developer to send a SQL statement directly to the DRDA server without the statement being interpreted by Oracle database. `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` SQL passthrough statements that are supported by the gateway are limited to nonqueries (`INSERT`, `UPDATE`, `DELETE`, and DDL statements) and cannot contain bind variables. The gateway can run native SQL statements 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 data definition language (DDL) statements, the function returns zero.

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

The 11.1 release of Oracle Database Gateway for DRDA 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 Results Sets Through Passthrough" on page 4-25 for more information.

Processing DDL Statements through Passthrough

As noted above, SQL statements which are processed through the `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` function are not interpreted by the Oracle database server. As a result, the Oracle database server will not know if such statements are making any modifications to the DRDA server. This means that unless you keep the Oracle database's cached information up to date after changes to the DRDA server, the database may continue to rely upon inaccurate or outdated information in subsequent queries within the same session.

An example of this occurs when you alter the structure of a table by either adding or removing a column. When an application references a table through the gateway (for example, when you perform a query on it), the Oracle database server caches the table definition. Now, suppose that (within the same session) the application subsequently alters the table's form, by using `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` to add a column. Then, the next reference to the table (by the application) will return the old column definitions of the table and will ignore the table's new column. This is because the Oracle database server did not process the statement and, so, has no knowledge of the alteration. Because the database does not know of the alteration, it has no reason to requery the table form, and, so, it will use the already-cached form to handle any new queries.

In order for the Oracle database server to acquire the new form of the table, the existing session with the gateway must be closed and a new session must be opened. This can be accomplished in either of two ways:

- By ending the application session with the Oracle database server and starting a new session after modifications have been made to the DRDA server; or
- By running the `ALTER SESSION CLOSE DATABASE LINK` command after making any modifications to the DRDA server.

Either of the above actions will void the cached table definitions and will force the Oracle database server to acquire new definitions on the next reference.

Using DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE

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_DRDA_sql');
```

where:

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_DRDA_sql is a valid nonquery SQL statement (except CONNECT, COMMIT, and ROLLBACK). The statement cannot contain bind variables. Native SQL statements that cannot be dynamically prepared are rejected by the DRDA server. The SQL statement passed by the DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE function must be a character string. For more information regarding valid SQL statements, refer to the SQL Reference for the particular DRDA server.

Examples

1. 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;
/
```

2. 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 Results Sets Through Passthrough

Oracle Database Gateway for DRDA provides a facility to retrieve results sets from a SELECT SQL statement entered through passthrough. Refer to *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_VALUE@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;
/
```

Oracle Data Dictionary Emulation on a DRDA Server

The gateway optionally augments the DRDA database catalogs with data dictionary views modeled on the Oracle data dictionary. These views are based on the dictionary tables in the DRDA database, presenting that 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 users with an Oracle-like interface to the contents and use of the DRDA database. Some of these views are required by Oracle products. The gateway supports the DB2/OS390, DB2/400, and DB2/UDB catalog views.

You can query the gateway data dictionary views to see the objects in the DRDA database and to determine the authorized users of the DRDA database. Many Oracle catalog views are supported by the Oracle Database Gateway for DRDA. Refer to Appendix A for descriptions of Oracle DB2 catalog views. These views are completely compatible with the gateway.

Using the DRDA Catalog

Each DRDA database has its own catalog tables and views, which you might find useful. Refer to the appropriate IBM documentation for descriptions of these catalogs.

Defining the Number of DRDA Cursors

You can define any number of cursors depending on your application requirements. Oracle recommends that you use the default value of 100. However, if the default is not appropriate for your application, there are two points to consider when defining the number of cursors for your installation:

- Each cursor requires an additional amount of storage and additional management.
- If you change `DRDA_PACKAGE_SECTIONS`, you must rebind the package.

The parameter DRDA_PACKAGE_SECTIONS is specific to the DRDA package. This parameter defines the number of sections (open cursors at the IBM database). Refer to Appendix B, "Initialization Parameters" for more information about setting the DRDA_PACKAGE_SECTIONS parameter.

5

Error Messages, Diagnosis, and Reporting

This chapter provides information about error messages and error codes. This data is specific to the 11.1 release of the Oracle Database Gateway for DRDA. This chapter contains the following sections:

Interpreting Gateway Error Messages

The gateway architecture consists of different components. Any component may detect and report an error condition while processing SQL statements that refer to one or more DRDA database tables. This means that errors can be complex, involving error codes and supporting data from multiple components. In all cases, however, the application ultimately receives a single error code or a return code.

As most gateway messages exceed the 70 character message area in the Oracle SQL Communications Area (SQLCA), the programmatic interfaces and Oracle Call Interfaces, that you use to access data through the gateway should use SQLGLM or OERHMS to view the entire text of messages. Refer to the programmer's guide to the Oracle precompilers for additional information about SQLGLM, and refer to the *Oracle C++ Call Interface Programmer's Guide* for additional information about OERHMS.

Errors encountered when using the gateway can originate from many sources, as follows:

- Errors detected by the Oracle database
- Errors detected by the gateway
- Errors detected in the DRDA software, either on the client or server side
- Communication errors
- Errors detected by the server database

Errors Detected by the Oracle Database

Errors detected by the Oracle database are reported back to the application or tool with the standard ORA type message. Refer to *Oracle Database Error Messages* for descriptions of these errors. For example, the following error occurs when an undefined database link name is specified:

```
ORA-02019:  connection description for remote database not found
```

Errors in the ORA-9100 to ORA-9199 range are reserved for the generic gateway layer (components of the gateway that are not specific to DRDA). Messages in this range are documented in *Oracle Database Error Messages*.

Errors Detected by the Gateway

Errors detected by the generic gateway are prefixed with HGO- and are documented in *Oracle Database Error Messages*.

A sample error message is:

```
HGO-00706:  HGO:  Missing equal sign for parameter in initialization file.
```

Errors Detected in the DRDA Software

Errors detected in the DRDA gateway, on the client or server side, are usually reported with error ORA-28500, followed by a gateway-specific expanded error message. There are two return codes reported in the expanded message:

- drc specifies DRDA-specific errors that are documented in "Gateway Error Codes" on page 5-4.
- grc specifies generic gateway errors detected in the DRDA layer. These errors are documented in the *Oracle Database Error Messages*.

> **Note:** Error code ORA-28500 was error code ORA-09100 prior to gateway version 8. Error code ORA-28501 was listed as ORA-09101 prior to gateway version 8.

The values in parentheses that follow the drc values are used for debugging by Oracle Support Services. The errp field indicates the program (client or server) that detected the error. If present, errmc lists any error tokens.

For example, the following error message is returned when the database name specified with the DRDA_REMOTE_NAME parameter in the initsid.ora file is not defined at the DRDA server:

```
ORA-28500: connection from ORACLE to a non-Oracle system returned the message:
DG4DRDA v11.1.0.5.0 grc=0, drc=-30061 (839C,0000), errp=GDJRFS2E
errmc=XNAME
```

Communication Errors

Communication errors are reported with an ORA-28501 followed by a gateway specific error message with drc=-30080 or drc=30081 indicating either a network error or lost session condition. errmc indicates which network function encountered the error, followed by a network interface specific error number.

For example, the following error message is returned when there is a failure to establish a session because DRDA_CONNECT_PARM in the initsid.ora file specifies a Side Information Profile that is not defined:

```
ORA-28501: communication error on heterogeneous database link
DG4DRDA v11.1.0.5.0 grc=0, drc=-30081  (839C,0001), errp= file or directory(2)
errmc=Initialize_Conversation (CMINIT) CM_PROGRAM_PARAMETER_CHECK(24) No such >
file or directory(2)
```

Refer to the appropriate host operating system documentation for more information.

Errors Detected by the Server Database

Errors detected by the server database are reported with an ORA-28500 followed by a gateway-specific expanded error message with drc=-777 sqlcode follows.) This is followed by another line that contains the sqlcode, sqlstate, errd (error array), and errmc (error tokens) returned from the DRDA server database. Refer to IBM documentation for the specific database being used. Also refer to Mapped Errors in this chapter for some SQL errors that get translated.

> **Note:** Error code ORA-28500 was error code ORA-09100 prior to gateway version 8. Error code ORA-28501 was listed as ORA-09101 prior to gateway version 8.

For example, the following error message indicates that the DRDA server database did not recognize the collection ID or package name specified with the DRDA_PACKAGE_COLLID or DRDA_PACKAGE_NAME parameters in the init*sid*.ora file:

```
ORA-28500: connection from ORACLE to a non-Oracle system returned the message:
DG4DRDA v11.1.0.5.0 grc=0, drc=-30020 (839C,0000), errp=GDJMRCM
sqlcode=-805, sqlstate=51002, errd=FFFFFF9C,0,0,FFFFFFFF,0,0
errmc=124c
```

Mapped Errors

Some SQL errors are returned from the DRDA server database and are translated to an Oracle error code. This is needed when the Oracle instance or gateway provides special handling of an error condition. The following table lists the mapped SQLstate error numbers, descriptions, and their corresponding Oracle error codes:

Table 5–1 Mapped sqlstate Errors

Description	sqlstate error	Oracle error
No rows selected	02000	0
Unique index constraint violated	23505	ORA-00001
Object does not exist	52004 or 42704	ORA-00942
Object name too long (more than 18 characters), and therefore object does not exist	54003 or 42622	ORA-00942
Insufficient privileges	42501	ORA-01031
Invalid CCSID (unimplemented character set conversion)	22522	ORA-01460
Invalid username/password; logon denied	N/A	ORA-01017
Divide by zero error	01519 or 01564	ORA-01476

The following is an example of a translated object does not exist error:

```
ORA-00942: table or view does not exist
DG4DRDA v11.1.0.5.0 grc=0, drc=-942 (839C,0001), errp=DSNXEDST
sqlcode=-204, sqlstate=52004, errd=32,0,0,FFFFFFFF,0,0
```

```
errmc=AJONES.CXDCX
```

Gateway Error Codes

Listed below are the common Oracle Database Gateway for DRDA error codes that appear in the `drc=` field of the expanded error messages. If you obtain a `drc` value that does not appear here, then contact Oracle Support Services.

-700 Invalid ORA_MAX_DATE specified

Cause: An invalid value was specified for `ORA_MAX_DATE` in the `init`*sid*`.ora` file.

Action: Correct the value of `ORA_MAX_DATE`. The correct format is `ORA_MAX_DATE=YYYY-MM-DD`, where `MM` is in the range of 1 to 12, and `DD` is in the range of 1 to 31 (and must be valid for the month).

-701 Default CCSID value not supported

Cause: The value specified for `DRDA_DEFAULT_CCSID` in the `init`*sid*`.ora` file is not supported by the Oracle Database Gateway for DRDA.

Action: Refer to Appendix C, "Globalization Support for DRDA", for a list of supported DRDA server character sets.

-702 Application Host (bind) variable exceeds 32K

Cause: An application program specified a host variable with length greater than the DRDA allowed maximum of 32K.

Action: The application must be modified to take into account DRDA limits.

-703 Local character set not supported

Cause: The character set specified for the `LANGUAGE` parameter in the `init`*sid*`.ora` file is not supported.

Action: Refer to Appendix C, "Globalization Support for DRDA", for a list of supported character sets.

-704 UserID length greater than maximum

Cause: The user ID used for the logon by the gateway is longer than 8 characters.

Action: A user ID of length of 8 or less must be used. Refer to Chapter 15, "Security Considerations" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* for a discussion of user IDs.

-705 Password length greater than maximum

Cause: The password being used for the logon by the gateway is longer than 8 characters.

Action: A password of length of 8 or less must be used. Refer to Chapter 15, "Security Considerations" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* for a discussion of passwords.

-777 DRDA Server RDBMS (SQL) Error

Cause: Server database detected an application-level SQL error.

Action: Refer to "Interpreting Gateway Error Messages" on page 5-1. `sqlcode` and, for more information to fix your application.

-30060 Invalid UserID/Password (DRDA Server RDBMS Authorization)

Cause: You have used a user ID or password that is not accepted by the DRDA server database.

Action: Refer to Chapter 15, "Security Considerations" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* or *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows* for user ID/password considerations.

-30061 RDB not found

Cause: The remote database specified with the `DRDA_REMOTE_DB_NAME` parameter is not a valid database in the DRDA server.

Action: Correct the value of the `DRDA_REMOTE_DB_NAME` parameter in the `init`*`sid`*`.ora` file.

-30080 Communication Error

Cause: The gateway encountered a communication error.

Action: Try processing the received error. If it persists, then refer to "Interpreting Gateway Error Messages" on page 5-1 and report to your system administrator.

-30081 Communication Error - lost session

Cause: The current DRDA network session was disconnected.

Action: Try processing the received error. If it persists, then refer to "Interpreting Gateway Error Messages" on page 5-1 and report it to your system administrator.

SQL Tracing and the Gateway

When developing applications, it is often useful to be able to see the exact SQL statements that are being passed through the gateway. This section describes setting appropriate trace parameters and setting up the debug gateway.

SQL Tracing in the Oracle Database

Oracle database has a command for capturing the SQL statement which is actually sent to the gateway. This command is called `EXPLAIN PLAN`. `EXPLAIN PLAN` is used to determine the execution plan that Oracle database follows to execute a specified SQL statement. This command inserts a row (describing each step of the execution plan) into a specified table. If you are using cost-based optimization, then this command also determines the cost of executing the statement. The syntax of the command is:

```
EXPLAIN PLAN [ SET STATEMENT_ID = 'text' ]
    [ INTO [schema.]table[@dblink] ] FOR statement
```

For detailed information on this command, refer to the *Oracle Database SQL Language Reference*.

Note: In most cases, EXPLAIN PLAN should be sufficient to extract the SQL statement that is actually sent to the gateway, and thus sent to the DRDA server. However, certain SQL statement form have post-processing performed on them in the gateway.

SQL Tracing in the Gateway

For Unix Based Systems:

The production gateway does not have built-in tracing built into it for the purpose of enhancing its speed. The product ships with a debug library that can be used to build a debug gateway for the purposes of tracing and debugging applications.

First, login using the Admin user ID of the gateway and setup the environment:

```
$ su - <gateway-Admin-User>
```

Next, build the debug gateway:

```
$ cd $ORACLE_HOME/dg4drda/lib
$ make -f dg4drda.mk ORACLE_HOME=your_oracle_home g4drsrvd
```

This will create the debug gateway and store it in $ORACLE_HOME/bin/g4drsrvd. Next, change the listener.ora to invoke the debug gateway. Change the PROGRAM parameter to specify the debug program name:

```
(SID_DESC=
        (SID_NAME=drdahoa1)
        (ORACLE_HOME=/oracle/dg4drda/11.1.0)
        (PROGRAM=g4drsrvd))
```

The listener will need to be reloaded for this change to take effect. Next, edit the Gateway Initialization File and add the following parameters:

TRACE_LEVEL and

ORACLE_DRDA_TCTL

You may optionally add the LOG_DESTINATION parameter, but it is not required.

The following is a fragment of a Gateway Initialization File with the parameters set:

```
#
TRACE_LEVEL=255
ORACLE_DRDA_TCTL=debug.tctl
#
```

The above example provides full tracing of both gateway and DRDA. In many cases, only the gateway tracing is desirable. To obtain only gateway tracing, remove (or comment out) the "ORACLE_DRDA_TCTL" parameter.

If you specify a LOG_DESTINATION, then you may specify only the file name (for example, drda.trc), in which case the log will be written to the log directory of the gateway ($ORACLE_HOME/dg4drda/log). Or you may specify a fully qualified path name. If you do not specify a LOG_DESTINATION, then a unique log file in a default format will be generated.

The logfile name will be of the form:

```
gatewaysid_pid.trc
```

where:

gatewaysid is the SID of the gateway. The value of this is determined by the setting of the FDS_INSTANCE parameter, and pid is the process identifier (PID) of the gateway process.

An example log file name would be:

```
drdahoa1_3875.trc
```

When searching for the SQL statements that are passed to the DRDA server, look for the strings '*** HGAPARS ***' and '*** HGAXMSQL ***'. The string after HGAPARS will be the incoming statement from the Oracle Database 11*g* Relational database Management System (RDBMS). The string after HGAXMSQL will be the outgoing statement after any date substitution is done. This is the actual SQL statement which will be given to the DRDA server.

When you have developed your application, revert the PROGRAM= value in the listener.ora to its previous value and reload the listener to use the production gateway again. You should also comment out the trace parameters in the Gateway Initialization Files.

For Microsoft Windows:

To enhance speed of the gateway, tracing was not built into the production gateway.

The product ships with a debug version of the gateway for the purposes of tracing and debugging applications.

This process entails changing the listener.ora file to use the debug gateway:

1. Log in as the Administrator user ID of the gateway and set up the environment.
2. Stop the Oracle Net Listener:

   ```
   > lsnrctl stop
   ```

3. Edit the listener.ora with any text editor:

   ```
   > notepad C:\Oracle\GTWHome\network\admin\listener.ora
   ```

4. Find the TNS entry for the gateway and change the program this way:

   ```
   PROGRAM=g4drsrvd
   ```

5. Save the file and exit. Next, restart the Oracle Net Listener:

   ```
   > lsnrctl start
   ```

6. Edit the gateway's init*sid*.ora file with any text editor:

   ```
   > notepad C:\Oracle\GTWHome\dg4drda\admin\init<i>sid</i>.ora
   ```

7. Set the following parameters:

   ```
   TRACE_LEVEL=255
   ORACLE_DRDA_TCTL=debug.tctl
   ```

You may, as an option, add the LOG_DESTINATION parameter, but it is not required. If you specify a LOG_DESTINATION, then you may specify just the file name (for example, drda.trc), or you may specify a fully qualified path name. If you specify a LOG_DESTINATION with just the file name, then the log will be written to the log directory (ORACLE_HOME\dg4drda\trace) of the gateway. If you do not specify a LOG_DESTINATION, then a unique log file in a default format will be generated. The log file name will be of the form:

```
gatewaysid_tid.trc
```

Where:

gateway sid is the SID of the gateway.

tid is the thread identifier (TID) of the gateway service.

An example log file name would be:

```
drdahoa1_3875.trc
```

When searching for the SQL statements which are passed to the DRDA Server, look for the strings `'*** HGAPARS ***'` and `'*** HGAXMSQL ***'`. The string after `HGAPARS` will be the incoming statement from the Oracle Database 11g RDBMS. The string after `HGAXMSQL` will be the outgoing statement after any date substitution is done. This is the actual SQL statement which will be given to the DRDA Server.

When you have finished developing your application, revert the `PROGRAM=` value in the listener.ora file to its previous value and reload the listener to use the production gateway again. You should also comment out the trace parameters in the gateway initialization files.

A

Oracle DB2 Data Dictionary Views

This appendix covers the Oracle Database Gateway for DRDA data dictionary views accessible to all users of Oracle database. Most of the views can be accessed by any user with `SELECT` privileges for DB2 catalog tables.

N/A is used in the tables to denote that the column is not valid for the gateway.

This appendix contains the following sections:

Supported Views

The following is a list of Oracle data dictionary views that are supported by the gateway for DB2/OS390, DB2/400, and DB2/UDB DRDA servers.

- ALL_CATALOG
- ALL_COL_COMMENTS
- ALL_CONS_COLUMNS
- ALL_CONSTRAINTS
- ALL_INDEXES
- ALL_IND_COLUMNS
- ALL_OBJECTS
- ALL_SYNONYMS
- ALL_TAB_COMMENTS
- ALL_TABLES
- ALL_TAB_COLUMNS
- ALL_USERS
- ALL_VIEWS
- COL_PRIVILEGES
- DICTIONARY
- DUAL
- TABLE_PRIVILEGES
- USER_CATALOG

- USER_COL_COMMENTS
- USER_CONSTRAINTS
- USER_CONS_COLUMNS
- USER_INDEXES
- USER_OBJECTS
- USER_SYNONYMS
- USER_TABLES
- USER_TAB_COLUMNS
- USER_TAB_COMMENTS
- USER_USERS
- USER_VIEWS

Data Dictionary View Tables

This section contains tables that describes describing data dictionary views. In the following descriptions, all are supported for DB2/OS390 and DB2/400.

ALL_CATALOG

All tables, views, synonyms, and sequence accessible to the user:

Column name	Description
OWNER	Owner of the object
TABLE_NAME	Name of the object
TABLE_TYPE	Type of object

ALL_COL_COMMENTS

Comments on columns of accessible tables and views:

Column name	Description
OWNER	Owner of the object
TABLE_NAME	Object name
COLUMN_NAME	Column name
COMMENTS	Comments on column

ALL_CONS_COLUMNS

Information about accessible columns in constraint definitions:

Column name	Description
OWNER	Owner of the constraint definition
CONSTRAINT_NAME	Name of the constraint definition
TABLE_NAME	Name of the table with a constraint definition

Column name	Description
COLUMN_NAME	Name of the column specified in the constraint definition
POSITION	Original position of column in definition

ALL_CONSTRAINTS

Constraint definitions on accessible tables:

Column name	Description
OWNER	Owner of the constraint definition
CONSTRAINT_NAME	Name of the constraint definition
CONSTRAINT_TYPE	Type of the constraint definition
TABLE_NAME	Name of the table with constraint definition
SEARCH_CONDITION	Text of the search condition for table check
R_OWNER	Owner of the table used in referential constraint
R_CONSTRAINT_NAME	Name of the unique constraint definition for referenced table
DELETE_RULE	Delete rule for a referential constraint
STATUS	Status of a constraint
DEFERRABLE	Whether the constraint is deferrable
DEFERRED	Whether the constraint was initially deferred
VALIDATED	Whether all data obeys the constraint
GENERATED	Whether the name of the constraint is user or system generated
BAD	Constraint specifies a century in an ambiguous manner
RELY	Whether an enabled constraint is enforced or unenforced
LAST_CHANGE	When the constraint was last enabled
INDEX_OWNER	N/A
INDEX_NAME	N/A

ALL_INDEXES

Description of indexes on tables accessible to the user:

Column name	Description
OWNER	Owner of the index
INDEX_NAME	Name of the index
INDEX_TYPE	Type of the index
TABLE_OWNER	Owner of the indexed object
TABLE_NAME	Name of the indexed object
TABLE_TYPE	Type of the indexed object
UNIQUENESS	Uniqueness status of the index
COMPRESSION	N/A

Column name	Description
PREFIX_LENGTH	0
TABLESPACE_NAME	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_THRESHOLD	Threshold percentage of block space allowed per index entry
INCLUDE_COLUMN	Column ID of the last column to be included in an index-organized table
FREELISTS	Number of process freelists allocated to this segment
FREELIST_GROUPS	Number of freelist groups allocated to this segment
PCT_FREE	N/A
LOGGING	Logging information
BLEVEL	Depth of the index from its root block to its leaf blocks. A depth of 1 indicates that the root block and the leaf block are the same.
LEAF_BLOCKS	Number of leaf blocks in the index
DISTINCT_KEYS	Number of distinct indexed values. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is the same as the number of rows in the table.
AVG_LEAF_BLOCKS_PER_KEY	N/A
AVG_DATA_BLOCKS_PER_KEY	N/A
CLUSTERING_FACTOR	N/A
STATUS	State of the index: VALID
NUM_ROWS	Number of rows in the index
SAMPLE_SIZE	Size of the sample used to analyze the index
LAST_ANALYZED	Date on which an index was most recently analyzed
DEGREE	Number of threads per instance for scanning the index
INSTANCES	Number of instances across which the index is to be scanned
PARTITIONED	Whether the index is partitioned
TEMPORARY	Whether the index is on a temporary table
GENERATED	Whether the name of the index is system generated
SECONDARY	N/A
BUFFER_POOL	Whether the index is a secondary object
USER_STATS	N/A
DURATION	N/A

Column name	Description
PCT_DIRECT_ACCESS	N/A
ITYP_OWNER	N/A
ITYP_NAME	N/A
PARAMETERS	N/A
GLOBAL_STATS	N/A
DOMIDX_STATUS	N/A
DOMIDX_OPSTATUS	N/A
FUNCIDX_STATUS	N/A
JOIN_INDEX	N/A
IOT_REDUNDANT_PKEY_ELIM	N/A

ALL_IND_COLUMNS

ALL_IND_COLUMNS describes the columns of indexes on all tables that are accessible to the current user.

Column names	Description
INDEX_OWNER	Owner of the index
INDEX_NAME	Name of the index
TABLE_OWNER	Owner of the table or cluster
TABLE_NAME	Name of the table or cluster
COLUMN_NAME	Column name or attribute of object type column
COLUMN_POSITION	Position of a column or attribute within the index
COLUMN_LENGTH	Indexed length of the column
CHAR_LENGTH	Maximum codepoint length of the column
DESCEND	Whether the column is sorted in descending order (Y/N)

ALL_OBJECTS

Objects accessible to the user:

Column name	Description
OWNER	Owner of the object
OBJECT_NAME	Name of object
SUBOBJECT_NAME	Name of the subobject
OBJECT_ID	Object number of the object
DATA_OBJECT_ID	Dictionary object number of the segment that contains the object
OBJECT_TYPE	Type of object
CREATED	N/A
LAST_DDL_TIME	N/A
TIMESTAMP	N/A

Column name	Description
STATUS	State of the object
TEMPORARY	Whether the object is temporary
GENERATED	Whether the name of this object system is generated
SECONDARY	N/A

ALL_SYNONYMS

All synonyms accessible to the user:

Column name	Description
OWNER	Owner of the synonym
SYNONYM_NAME	Name of the synonym
TABLE_OWNER	Owner of the object referenced by the synonym
TABLE_NAME	Name of the object referenced by the synonym
DB_LINK	N/A

ALL_TABLES

Description of tables accessible to the user:

Column name	Description
OWNER	Owner of the table
TABLE_NAME	Name of the table
TABLESPACE_NAME	Name of the tablespace containing the table
CLUSTER_NAME	N/A
IOT_NAME	Name of the index organized table
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
FREELISTS	Number of process freelists allocated to this segment
FREELIST_GROUPS	Number of freelist groups allocated to this segment
LOGGING	Logging attribute
BACKED_UP	N/A
NUM_ROWS	Number of rows in the table
BLOCKS	N/A

Column name	Description
EMPTY_BLOCKS	N/A
AVG_SPACE	N/A
CHAIN_CNT	N/A
AVG_ROW_LEN	Average length of a row in the table in bytes
AVG_SPACE_FREELIST_BLOCKS	Average freespace of all blocks on a freelist
NUM_FREELIST_BLOCKS	Number of blocks on the freelist
DEGREE	Number of threads per instance for scanning the table
INSTANCES	Number of instances across which the table is to be scanned
CACHE	Whether the cluster is to be cached in the buffer cache
TABLE_LOCK	Whether the table locking is enabled or disabled
SAMPLE_SIZE	Sample size used in analyzing this table
LAST_ANALYZED	Date on which this table was most recently analyzed
PARTITIONED	Whether this table is partitioned
IOT_TYPE	Whether the table is an index-organized table
TEMPORARY	Can the current session only see data that it placed in this object itself?
SECONDARY	N/A
NESTED	Whether the table is a nested table
BUFFER_POOL	Default buffer pool for the object
ROW_MOVEMENT	N/A
GLOBAL_STATS	N/A
USER_STATS	N/A
DURATION	N/A
SKIP_CORRUPT	N/A
MONITORING	N/A
CLUSTER_OWNER	N/A
DEPENDENCIES	N/A
COMPRESSION	N/A

ALL_TAB_COLUMNS

Columns of all tables, views, and clusters accessible to the user:

Column name	Description
OWNER	Owner of the table or view
TABLE_NAME	Table or view name
COLUMN_NAME	Column name
DATA_TYPE	Data type of the column

Column name	Description
DATA_TYPE_MOD	Data type modifier of the column
DATA_TYPE_OWNER	Owner of the data type of the column
DATA_LENGTH	Maximum length of the column in bytes
DATA_PRECISION	N/A
DATA_SCALE	Digits to the right of decimal point in a number
NULLABLE	Whether the column permits nulls? 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	Sequence number of the column as created
DEFAULT_LENGTH	N/A
DATA_DEFAULT	N/A
NUM_DISTINCT	Number of distinct values in each column of the table
LOW_VALUE	For tables with more than three rows, the second lowest and second highest values. These statistics are expressed in hexadecimal notation for the internal representation of the first 32 bytes of the values.
HIGH_VALUE	N/A
DENSITY	N/A
NUM_NULLS	Number of nulls in the column
NUM_BUCKETS	Number of buckets in histogram for the column
LAST_ANALYZED	Date on which this column was most recently analyzed
SAMPLE_SIZE	Sample size used in analyzing this column
CHARACTER_SET_NAME	Name of the character set
CHAR_COL_DECL_LENGTH	Length of the character set
GLOBAL_STATS	N/A
USER_STATS	N/A
AVG_COL_LEN	Average length of the column (in bytes)
CHAR_LENGTH	Displays the length of the column in characters
CHAR_USED	N/A

ALL_TAB_COMMENTS

Comments on tables and views accessible to the user:

Column name	Description
OWNER	Owner of the object
TABLE_NAME	Name of the object
TABLE_TYPE	Type of the object
COMMENTS	Comments on the object

ALL_USERS

Information about all users of the database:

Column name	Description
USERNAME	Name of the user
USER_ID	N/A
CREATED	N/A

ALL_VIEWS

Text of views accessible to the user:

Column name	Description
OWNER	Owner of the view
VIEW_NAME	Name of the view
TEXT_LENGTH	Length of the view text
TEXT	View text. Only the first row of text is returned, even if multiple rows exist.
TYPE_TEXT_LENGTH	Length of the type clause of the typed view
TYPE_TEXT	Type clause of the typed view
OID_TEXT_LENGTH	Length of the WITH OID clause of the typed view
OID_TEXT	WITH OID clause of the typed view
VIEW_TYPE_OWNER	Owner of the type of the view, if the view is a typed view
VIEW_TYPE	Type of the view, if the view is a typed view
SUPERVIEW_NAME	N/A

COLUMN_PRIVILEGES

Grants on columns for which the user is the grantor, grantee, or owner, or PUBLIC is the grantee:

Column name	Description
GRANTEE	Name of the user to whom access was granted
OWNER	Username of the owner of the object
TABLE_NAME	Name of the object
COLUMN_NAME	Name of the column
GRANTOR	Name of the user who performed the grant
INSERT_PRIV	Permission to insert into the column
UPDATE_PRIV	Permission to update the column
REFERENCES_PRIV	Permission to reference the column
CREATED	Timestamp for the grant

DICTIONARY

List or data dictionary tables:

Column name	Description
TABLE_NAME	Table name
COMMENTS	Description of the table

DUAL

List of Dual tables:

Column name	Description
DUMMY	A dummy column

TABLE_PRIVILEGES

Grants on objects for which the user is the grantor, grantee, or owner, or PUBLIC is the grantee:

Column name	Description
GRANTEE	Name of the user to whom access is granted
OWNER	Owner of the object
TABLE_NAME	Name of the object
GRANTOR	Name of the user who performed the grant
SELECT_PRIV	Permission to select data from an object
INSERT_PRIV	Permission to insert data into an object
DELETE_PRIV	Permission to delete data from an object
UPDATE_PRIV	Permission to update an object
REFERENCES_PRIV	N/A
ALTER_PRIV	Permission to alter an object
INDEX_PRIV	Permission to create or drop an index on an object
CREATED	Timestamp for the grant

USER_CATALOG

Tables, views, synonyms, and sequences owned by the use:

Column name	Description
TABLE_NAME	Name of the object
TABLE_TYPE	Type of the object

USER_COL_COMMENTS

Comments on columns of user's tables and views:

Column name	Description
TABLE_NAME	Name of the object
COLUMN_NAME	Name of the column

Column name	Description
COMMENTS	Comments on the column

USER_CONSTRAINTS

Constraint definitions on user's tables:

Column name	Description
OWNER	Owner of the constraint definition
CONSTRAINT_NAME	Name associated with the constraint definition
CONSTRAINT_TYPE	Type of the constraint definition
TABLE_NAME	Name associated with the table with constraint definition
SEARCH_CONDITION	Text of the search condition for table check
R_OWNER	Owner of table used in referential constraint
R_CONSTRAINT_NAME	Name of the unique constraint definition for referenced table
DELETE_RULE	Delete rule for referential constraint
STATUS	Status of a constraint
DEFERRABLE	Whether the constraint is deferrable
DEFERRED	Whether the constraint was initially deferred
VALIDATED	Whether all data obeys the constraint
GENERATED	Whether the name of the constraint is user or system generated
BAD	Constraint specifies a century in an ambiguous manner
LAST_CHANGE	When the constraint was last enabled
INDEX_OWNER	N/A
INDEX_NAME	N/A

USER_CONS_COLUMNS

Information about columns in constraint definitions owned by the user:

Column name	Description
OWNER	Owner of the constraint definition
CONSTRAINT_NAME	Name associated with the constraint definition
TABLE_NAME	Name associated with table with constraint definition
COLUMN_NAME	Name associated with column specified in the constraint definition
POSITION	Original position of column in definition

USER_INDEXES

Description of the user's own indexes:

Column name	Description
INDEX_NAME	Name of the index
INDEX_TYPE	Type of index
TABLE_OWNER	Owner of the indexed object
TABLE_NAME	Name of the indexed object
TABLE_TYPE	Type of the indexed object
UNIQUENESS	Uniqueness status of the index
COMPRESSION	N/A
PREFIX_LENGTH	0
TABLESPACE_NAME	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_THRESHOLD	Threshold percentage of block space allowed per index entry
INCLUDE_COLUMN	Column ID of the last column to be included in index-organized table
FREELISTS	Number of process freelists allocated to a segment
FREELIST_GROUPS	Number of freelist groups allocated to a segment
PCT_FREE	N/A
LOGGING	Logging information
BLEVEL	Depth of the index from its root block to its leaf blocks. A depth of 1 indicates that the root and leaf block are the same.
LEAF_BLOCKS	Number of leaf blocks in the index
DISTINCT_KEYS	Number of distinct indexed values. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is the same as the number of rows in the table.
AVG_LEAF_BLOCKS_PER_KEY	N/A
AVG_DATA_BLOCKS_PER_KEY	N/A
CLUSTERING_FACTOR	N/A
STATUS	State of the indexes: VALID
NUM_ROWS	Number of rows in the index
SAMPLE_SIZE	Size of the sample used to analyze the index
LAST_ANALYZED	Date on which the index was most recently analyzed
DEGREE	Number of threads per instance for scanning the index

Column name	Description
INSTANCES	Number of instances across which the index is to be scanned
PARTITIONED	Whether the index is partitioned
TEMPORARY	Whether the index is on a temporary table
GENERATED	Whether the name of the index is system generated
SECONDARY	N/A
BUFFER_POOL	Whether the index is a secondary object
USER_STATS	N/A
DURATION	N/A
PCT_DIRECT_ACCESS	N/A
ITYP_OWNER	N/A
ITYP_NAME	N/A
PARAMETERS	N/A
GLOBAL_STATS	N/A
DOMIDX_STATUS	N/A
DOMIDX_OPSTATUS	N/A
FUNCIDX_STATUS	N/A
JOIN_INDEX	N/A
IOT_REDUNDANT_PKEY_ELIM	N/A

USER_OBJECTS

Objects owned by the user:

Column name	Description
OBJECT_NAME	Name of the object
SUBOBJECT_NAME	Name of the subobject
OBJECT_ID	Object number of the object
DATA_OBJECT_ID	Dictionary object number of the segment that contains the object
OBJECT_TYPE	Type of object
CREATED	N/A
LAST_DDL_TIME	N/A
TIMESTAMP	N/A
STATUS	State of the object: VALID
TEMPORARY	Whether the object is temporary
GENERATED	Was the name of this object system generated?
SECONDARY	N/A

USER_SYNONYMS

The user's private synonyms:

Column name	Description
SYNONYM_NAME	Name of the synonym
TABLE_OWNER	Owner of the object referenced by the synonym
TABLE_NAME	Name of the object referenced by the synonym
DB_LINK	N/A

USER_TABLES

Description of the user's own tables:

Column name	Description
TABLE_NAME	Name of the table
TABLESPACE_NAME	Name of the tablespace containing the table
CLUSTER_NAME	N/A
IOT_NAME	Name of the index organized table
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
FREELISTS	Number of process freelists allocated to a segment
FREELIST_GROUPS	Number of freelist groups allocated to a segment
LOGGING	Logging information
BACKED_UP	N/A
NUM_ROWS	Number of rows in the table
BLOCKS	N/A
EMPTY_BLOCKS	N/A
AVG_SPACE	N/A
CHAIN_CNT	N/A
AVG_ROW_LEN	Average length of a row in the table in bytes
AVG_SPACE_FREELIST_BLOCKS	Average freespace of all blocks on a freelist
NUM_FREELIST_BLOCKS	Number of blocks on the freelist
DEGREE	Number of threads per instance for scanning the table
INSTANCES	Number of instances across which the table is to be scanned
CACHE	Whether the cluster is to be cached in the buffer cache

Column name	Description
TABLE_LOCK	Whether table locking is enabled or disabled
SAMPLE_SIZE	Sample size used in analyzing this table
LAST_ANALYZED	Date on which this table was most recently analyzed
PARTITIONED	Indicates whether this table is partitioned
IOT_TYPE	If this is an index organized table
TEMPORARY	Can the current session only see data that it placed in this object itself?
SECONDARY	N/A
NESTED	If the table is a nested table
BUFFER_POOL	The default buffer pool for the object
ROW_MOVEMENT	N/A
GLOBAL_STATS	N/A
USER_STATS	N/A
DURATION	N/A
SKIP_CORRUPT	N/A
MONITORING	N/A
CLUSTER_OWNER	N/A
DEPENDENCIES	N/A
COMPRESSION	N/A

USER_TAB_COLUMNS

Columns of user's tables, views, and clusters:

Column name	Description
TABLE_NAME	Name of the table, view, or cluster
COLUMN_NAME	Name of the column
DATA_TYPE	Data type of column
DATA_TYPE_MOD	Data type modifier of the column
DATA_TYPE_OWNER	Owner of the data type of the column
DATA_LENGTH	Maximum length of the column in bytes
DATA_PRECISION	N/A
DATA_SCALE	Digits to the right of a decimal point in a number
NULLABLE	Whether the column permits nulls. 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	Sequence number of the column as created
DEFAULT_LENGTH	N/A
DATA_DEFAULT	N/A
NUM_DISTINCT	Number of distinct values in each column of the table

Column name	Description
LOW_VALUE	For tables with more than three rows, the second lowest and second highest values. These statistics are expressed in hexadecimal notation for the internal representation of the first 32 bytes of the values.
HIGH_VALUE	N/A
DENSITY	N/A
NUM_NULLS	Number of nulls in the column
NUM_BUCKETS	Number of buckets in histogram for the column
LAST_ANALYZED	Date on which this column was most recently analyzed
SAMPLE_SIZE	Sample size used in analyzing this column
CHARACTER_SET_NAME	Name of the character set
CHAR_COL_DECL_LENGTH	Length of the character set
GLOBAL_STATS	N/A
USER_STATS	N/A
AVG_COL_LEN	Average length of the column (in bytes)
CHAR_LENGTH	Length of the column in characters
CHAR_USED	N/A

USER_TAB_COMMENTS

Comments on the tables and views owned by the user:

Column name	Description
TABLE_NAME	Name of the object
TABLE_TYPE	Type of the object
COMMENTS	Comments on the object

USER_USERS

Information about the current user:

Column name	Description
USERNAME	Name of the user
USER_ID	N/A
ACCOUNT_STATUS	Indicates if the account is locked, expired or unlocked
LOCK_DATE	Date on which the account was locked
EXPIRE_DATE	Date of expiration of the account
DEFAULT_TABLESPACE	N/A
TEMPORARY_TABLESPACE	N/A
CREATED	N/A
EXTERNAL_NAME	Name of the external user

USER_VIEWS

Text of views owned by the user:

Column name	Description
VIEW_NAME	Name of the view
TEXT_LENGTH	Length of the view text
TEXT	First line of the view text
TYPE_TEXT_LENGTH	Length of the type clause of the typed view
TYPE_TEXT	Type clause of the typed view
OID_TEXT_LENGTH	Length of the WITH OID clause of the typed view
OID_TEXT	WITH OID clause of the typed view
VIEW_TYPE_OWNER	Owner of the type of the view, if the view is a typed view
VIEW_TYPE	Type of the view, if the view is a typed view
SUPERVIEW_NAME	N/A

B

Initialization Parameters

The Oracle database initialization parameters in the `init.ora` file are distinct from gateway initialization parameters. Set the gateway parameters in the initialization parameter file using an agent-specific mechanism, or set them in the Oracle data dictionary using the `DBMS_HS` package. The gateway initialization parameter file must be available when the gateway is started. Changes made to the initialization parameters only take effect in the next gateway session.

This appendix contains a list of the gateway initialization parameters that can be set for each gateway and their description. It also describes the initialization parameter file syntax. It includes the following sections:

Initialization Parameter File Syntax

The syntax for the initialization parameter file is as follows:

1. The file is a sequence of commands.
2. Each command should start on a separate line.
3. End of line is considered a command terminator (unless escaped with a backslash).
4. If there is a syntax error in an initialization parameter file, none of the settings take effect.
5. Set the parameter values as follows:

   ```
   [SET][PRIVATE] parameter=value
   ```

   Where:

   `parameter` is an initialization parameter name. It is a string of characters starting with a letter and consisting of letters, digits and underscores. Initialization parameter names are case sensitive.

   `value` is the initialization parameter value. It is case-sensitive. An initialization parameter value is either:

   a. A string of characters that does not contain any backslashes, white space or double quotation marks (")

b. A quoted string beginning with a double quotation mark and ending with a double quotation mark. The following can be used inside a quoted string:

* backslash (\) is the escape character
* \n inserts a new line
* \t inserts a tab
* \" inserts a double quotation mark
* \\ inserts a backslash

A backslash at the end of the line continues the string on the next line. If a backslash precedes any other character then the backslash is ignored.

For example, to enable tracing for an agent, set the HS_FDS_TRACE_LEVEL initialization parameter as follows:

```
HS_FDS_TRACE_LEVEL=ON
```

SET and PRIVATE are optional keywords. You cannot use either as an initialization parameter name. Most parameters are needed only as initialization parameters, so you usually do not need to use the SET or PRIVATE keywords. If you do not specify either SET or PRIVATE, the parameter is used only as an initialization parameter for the agent.

SET specifies that, in addition to being used as an initialization parameter, the parameter value is set as an environment variable for the agent process. Use SET for parameter values that the drivers or non-Oracle system need as environment variables.

PRIVATE specifies that the initialization parameter should be private to the agent and should not be uploaded to the Oracle database. Most initialization parameters should not be private. If, however, you are storing sensitive information like a password in the initialization parameter file, then you may not want it uploaded to the server because the initialization parameters and values are not encrypted when uploaded. Making the initialization parameters private prevents the upload from happening and they do not appear in dynamic performance views. Use PRIVATE for the initialization parameters only if the parameter value includes sensitive information such as a username or password.

SET PRIVATE specifies that the parameter value is set as an environment variable for the agent process and is also private (not transferred to the Oracle database, not appearing in dynamic performance views or graphical user interfaces).

Oracle Database Gateway for DRDA Initialization Parameters

This section lists all the initialization file parameters that can be set for the Oracle Database Gateway for DRDA. They are as follows:

- HS_RPC_FETCH_REBLOCKING
- HS_RPC_FETCH_SIZE
- HS_TRANSACTION_MODEL
- HS_FDS_FETCH_ROWS
- IFILE
- DRDA_CACHE_TABLE_DESC
- DRDA_CAPABILITY
- DRDA_CODEPAGE_MAP
- DRDA_COMM_BUFLEN
- DRDA_CONNECT_PARM
- DRDA_DEFAULT_CCSID
- DRDA_DESCRIBE_TABLE
- DRDA_DISABLE_CALL
- DRDA_FLUSH_CACHE
- DRDA_GRAPHIC_CHAR_SIZE
- DRDA_GRAPHIC_PAD_SIZE
- DRDA_GRAPHIC_LIT_CHECK
- DRDA_GRAPHIC_TO_MBCS
- DRDA_ISOLATION_LEVEL
- DRDA_LOCAL_NODE_NAME
- DRDA_MBCS_TO_GRAPHIC
- DRDA_OPTIMIZE_QUERY
- DRDA_PACKAGE_COLLID
- DRDA_PACKAGE_CONSTOKEN
- DRDA_PACKAGE_NAME
- DRDA_PACKAGE_OWNER
- DRDA_PACKAGE_SECTIONS
- DRDA_READ_ONLY
- DRDA_RECOVERY_PASSWORD
- DRDA_RECOVERY_USERID
- DRDA_REMOTE_DB_NAME
- FDS_CLASS
- HS_NLS_NCHAR
- LOG_DESTINATION
- ORA_MAX_DATE
- ORA_NLS11
- ORACLE_DRDA_TCTL

Initialization Parameter Description

The following sections describe all the initialization file parameters that can be set for gateways.

HS_CALL_NAME

Property	Description
Default value	None
Range of values	Not applicable

Specifies the remote functions that can be referenced in SQL statements. The value is a list of remote functions and their owners, separated by semicolons, in the following format:

owner_name.function_name

For example:

```
owner1.A1;owner2.A2;owner3.A3
```

If an owner name is not specified for a remote function, the default owner name becomes the user name used to connect to the remote database (specified when the Heterogeneous Services database link is created or taken from user session if not specified in the DB link).

The entries for the owner names and the function names are case-sensitive.

HS_DB_DOMAIN

Property	Description
Default value	`WORLD`
Range of values	1 to 199 characters

Specifies a unique network sub-address for a non-Oracle system. The `HS_DB_DOMAIN` initialization parameter is similar to the `DB_DOMAIN` initialization parameter, described in the *Oracle Database Reference*. The `HS_DB_DOMAIN` initialization parameter is required if you use the Oracle Names server. The `HS_DB_NAME` and `HS_DB_DOMAIN` initialization parameters define the global name of the non-Oracle system.

> **Note:** The `HS_DB_NAME` and `HS_DB_DOMAIN` initialization parameters must combine to form a unique address in a cooperative server environment.

HS_DB_INTERNAL_NAME

Property	Description
Default value	01010101
Range of values	1 to 16 hexadecimal characters

Specifies a unique hexadecimal number identifying the instance to which the Heterogeneous Services agent is connected. This parameter's value is used as part of a transaction ID when global name services are activated. Specifying a nonunique number can cause problems when two-phase commit recovery actions are necessary for a transaction.

HS_DB_NAME

Property	Description
Default value	HO
Range of values	1 to 8 characters

Specifies a unique alphanumeric name for the data store given to the non-Oracle system. This name identifies the non-Oracle system within the cooperative server environment. The HS_DB_NAME and HS_DB_DOMAIN initialization parameters define the global name of the non-Oracle system.

HS_DESCRIBE_CACHE_HWM

Property	Description
Default value	100
Range of values	1 to 4000

Specifies the maximum number of entries in the describe cache used by Heterogeneous Services. This limit is known as the describe cache high water mark. The cache contains descriptions of the mapped tables that Heterogeneous Services reuses so that it does not have to re-access the non-Oracle data store.

If you are accessing many mapped tables, increase the high water mark to improve performance. Increasing the high water mark improves performance at the cost of memory usage.

HS_LANGUAGE

Property	Description
Default value	System-specific
Range of values	Any valid language name (up to 255 characters)

Provides Heterogeneous Services with character set, language, and territory information of the non-Oracle data source. The value must use the following format:

language[*_territory.character_set*]

Note: The globalization support initialization parameters affect error messages, the data for the SQL Service, and parameters in distributed external procedures.

Character Sets

Ideally, the character sets of the Oracle database and the non-Oracle data source are the same. If they are not the same, Heterogeneous Services attempts to translate the character set of the non-Oracle data source to the Oracle database character set, and back again. The translation can degrade performance. In some cases, Heterogeneous Services cannot translate a character from one character set to another.

Note: The specified character set must be a superset of the operating system character set on the platform where the agent is installed.

Language

The language component of the HS_LANGUAGE initialization parameter determines:

- Day and month names of dates
- AD, BC, PM, and AM symbols for date and time
- Default sorting mechanism

Note that Oracle does not determine the language for error messages for the generic Heterogeneous Services messages (ORA-25000 through ORA-28000). These are controlled by the session settings in the Oracle database.

Note: Use the HS_NLS_DATE_LANGUAGE initialization parameter to set the day and month names, and the AD, BC, PM, and AM symbols for dates and time independently from the language.

Territory

The territory clause specifies the conventions for day and week numbering, default date format, decimal character and group separator, and ISO and local currency symbols. Note that the level of globalization support between the Oracle database and the non-Oracle data source depends on how the gateway is implemented.

HS_OPEN_CURSORS

Property	Description
Default value	50
Range of values	1 to the value of OPEN_CURSORS initialization parameter of Oracle database

Defines the maximum number of cursors that can be open on one connection to a non-Oracle system instance.

The value never exceeds the number of open cursors in the Oracle database. Therefore, setting the same value as the OPEN_CURSORS initialization parameter in the Oracle database is recommended.

HS_RPC_FETCH_REBLOCKING

Property	Description
Default value	ON
Range of values	OFF or ON

Controls whether Heterogeneous Services attempts to optimize performance of data transfer between the Oracle database and the Heterogeneous Services agent connected to the non-Oracle data store.

The following values are possible:

- OFF disables reblocking of fetched data so that data is immediately sent from agent to server.
- ON enables reblocking, which means that data fetched from the non-Oracle system is buffered in the agent and is not sent to the Oracle database until the amount of fetched data is equal or higher than the value of HS_RPC_FETCH_SIZE initialization parameter. However, any buffered data is returned immediately when a fetch indicates that no more data exists or when the non-Oracle system reports an error.

HS_RPC_FETCH_SIZE

Property	Description
Default value	50000
Range of values	1 to 10000000

Tunes internal data buffering to optimize the data transfer rate between the server and the agent process.

Increasing the value can reduce the number of network round-trips needed to transfer a given amount of data, but also tends to increase data bandwidth and to reduce latency as measured between issuing a query and completion of all fetches for the query. Nevertheless, increasing the fetch size can increase latency for the initial fetch results of a query, because the first fetch results are not transmitted until additional data is available.

HS_TRANSACTION_MODEL

Property	Description
Default Value	COMMIT_CONFIRM
Range of Values	COMMIT_CONFIRM, READ_ONLY, SINGLE_SITE

Specifies the type of transaction model that is used when the non-Oracle database is updated by a transaction.

The following values are possible:

- COMMIT_CONFIRM provides read and write access to the non-Oracle database and allows the gateway to be part of a distributed update. To use the commit-confirm model, the following items must be created in the non-Oracle database:

  - Transaction log table. The default table name is HS_TRANSACTION_LOG. A different name can be set using the HS_FDS_TRANSACTION_LOG parameter. The transaction log table must be granted SELECT, DELETE, and INSERT privileges set to public.
  - Recovery account. The account name is assigned with the HS_FDS_RECOVERY_ACCOUNT parameter.
  - Recovery account password. The password is assigned with the HS_FDS_RECOVERY_PWD parameter.

    COMMIT_CONFIRM does not apply to Oracle Database Gateway for ODBC. The default value for Oracle Database Gateway for ODBC is SINGLE_SITE.

- READ_ONLY provides read access to the non-Oracle database.
- SINGLE_SITE provides read and write access to the non-Oracle database. However, the gateway cannot participate in distributed updates.

HS_FDS_FETCH_ROWS

Property	Description
Default Value	100
Range of Values	Any integer between 1 and 1000
Syntax	HS_FDS_FETCH_ROWS=*num*

HS_FDS_FETCH_ROWS specifies the fetch array size. This is the number of rows to be fetched from the non-Oracle database and to return to Oracle database at one time. This parameter will be affected by the HS_RPC_FETCH_SIZE and HS_RPC_FETCH_REBLOCKING parameters.

IFILE

Property	Description
Default value	None
Range of values	Valid parameter file names

Use the IFILE initialization parameter to embed another initialization file within the current initialization file. The value should be an absolute path and should not contain environment variables. The three levels of nesting limit does not apply.

See Also: *Oracle Database Reference*

DRDA_CACHE_TABLE_DESC

Property	Description
Default Value	TRUE
Range of Values	{TRUE|FALSE}
Syntax	DRDA_CACHE_TABLE_DESC=*{TRUE|FALSE}*

DRDA_CACHE_TABLE_DESC directs the gateway to cache table descriptions once per transaction. This can reduce the number of table lookups requested by Oracle database and can speed up the execution of SQL statements. You may wish to disable this option if you would be altering the structure of a remote table and if you would be examining it within the same transaction.

DRDA_CAPABILITY

Property	Description
Default Value	None
Range of Values	Refer to Chapter 4, "Developing Applications"
Syntax	DRDA_CAPABILITY={*FUNCTION/{ON|OFF}*},...

DRDA_CAPABILITY specifies which mapped functions of Oracle database will be treated natively. In other words, no special pre processing or post processing will be done for these functions. They will be passed through to the DRDA Server unmodified.

DRDA_CODEPAGE_MAP

Property	Description
Default Value	codepage.map
Range of Values	Any valid file path
Syntax	DRDA_CODEPAGE_MAP=*codepage.map*

DRDA_CODEPAGE_MAP specifies the location of the codepage map. You may specify only the filename, which will be searched for within the $ORACLE_HOME/dg4drda/admin directory, or you may specify the full path name of the file.

DRDA_COMM_BUFLEN

Property	Description
Default Value	32767
Range of Values	512 through 32767
Syntax	DRDA_COMM_BUFLEN=*num*

DRDA_COMM_BUFLEN specifies the communications buffer length. This is a number indicating the TCP/IP buffer size in bytes.

DRDA_CONNECT_PARM

Property	Description
Default Value	DRDACON1:446
Range of Values	Any alphanumeric string 1 to 255 characters in length

Property	Description
Syntax	DRDA_CONNECT_PARM={*hostname*|*ip_address*}{*:port*}

DRDA_CONNECT_PARM specifies the TCP/IP hostname or IP Address of the DRDA Server and, as an option, the Service Port number on which the DRDA Server is listening.

The DRDA standard specifies that port 446 be used for DRDA services. However, if several DRDA servers are operating on the same system, then they will need to provide service on different ports. Therefore, the port number that is used by each DRDA server will need to be extracted from the configuration of each individual DRDA server. DB2 for OS/390 and DB2/400 typically use the DRDA standard port number, 446, whereas DB2/UDB typically uses 50000 as the port number. Refer to IBM DB2 Administrator and Installation guides for locating and changing these port numbers for your DRDA server. For additional information, consult your DB2 DBA or System Administrator.

DRDA_DEFAULT_CCSID

Property	Description
Default Value	None
Range of Values	Any supported DRDA Server CCSID
Syntax	DRDA_DEFAULT_CCSID=*ccsid*

DRDA_DEFAULT_CCSID specifies the default CCSID or character set codepage for character set conversions when the DRDA Server database indicates that a character string has a CCSID of 65535. DRDA Servers use CCSID 65535 for columns specified as "FOR BIT DATA". In most cases, this parameter should not be specified, allowing CCSID 65535 to be treated as an Oracle RAW data type.

This parameter is for supporting databases (in particular, DB2/400) that use CCSID 65535 as the default for all tables created. Allowing CCSID 65535 to be treated as another CCSID can save such sites from having to modify every table.

> **Warning: Specifying any value for DRDA_DEFAULT_CCSID causes all "FOR BIT DATA" columns to be handled as text columns that need character set conversion and, therefore, any binary data in these columns can encounter conversion errors (ORA-28527).**

DRDA_DESCRIBE_TABLE

Property	Description
Default Value	TRUE
Range of Values	{TRUE|FALSE}
Syntax	DRDA_DESCRIBE_TABLE={*TRUE*|*FALSE*}

DRDA_DESCRIBE_TABLE directs the gateway to use the DRDA operation Table Describe to return the description of tables. This is an optimization that reduces the amount of time and resources that are used to look up the definition of a table.

Note: This feature is not compatible with DB2 Aliases or Synonyms. If you use DB2 aliases, then be sure to disable this option.

DRDA_DISABLE_CALL

Property	Description
Default Value	TRUE
Range of Values	{TRUE|FALSE}
Syntax	DRDA_DISABLE_CALL={*TRUE*|*FALSE*}

DRDA_DISABLE_CALL controls stored procedure usage, and is also used to control how the package is bound on the target database. This parameter should be set to FALSE only for supported target DRDA servers and should be set to TRUE otherwise.

Note: Any change to this parameter requires you to rebind.

DRDA_FLUSH_CACHE

Property	Description
Default Value	SESSION
Range of Values	{SESSION|COMMIT}
Syntax	DRDA_FLUSH_CACHE={*SESSION*|*COMMIT*}

DRDA_FLUSH_CACHE specifies when the cursor cache is to be flushed. With DRDA_FLUSH_CACHE=COMMIT, the cursor cache is flushed whenever the transaction is committed. With DRDA_FLUSH_CACHE=SESSION, the cache is not flushed until the session terminates.

DRDA_GRAPHIC_CHAR_SIZE

Property	Description
Default Value	4
Range of Values	1 through 4
Syntax	DRDA_GRAPHIC_CHAR_SIZE=num

DRDA_GRAPHIC_CHAR_SIZE is used to define the character conversion size to be used for GRAPHIC data types. It is a tuning parameter which affects the maximum size of a GRAPHIC data type when the column is described.

DRDA_GRAPHIC_PAD_SIZE

Property	Description
Default Value	0
Range of Values	0 through 127
Syntax	DRDA_GRAPHIC_PAD_SIZE=num

DRDA_GRAPHIC_PAD_SIZE is used to pad the size of a Graphic column as described by the DRDA Server. This is sometimes necessary depending upon the character set of the DRDA database and Oracle database. If the Oracle database is based on EBCDIC and the DRDA database is based on ASCII, then a pad size of 2 may be needed.

DRDA_GRAPHIC_LIT_CHECK

Property	Description
Default Value	FALSE
Range of Values	{TRUE|FALSE}
Syntax	DRDA_GRAPHIC_LIT_CHECK={*TRUE*|*FALSE*}

DRDA_GRAPHIC_LIT_CHECK directs the gateway to evaluate string literals within INSERT SQL statements in order to determine if they need to be converted to double-byte format for insertion into a Graphic column at the DRDA Server database. This is done by querying the column attributes of the table in the SQL statement to determine if a string literal is being applied to a column with a Graphic data type. If the table column is Graphic, and if this parameter is TRUE, then the gateway will rewrite the SQL statement with the literal converted to double-byte format. Existing double-byte characters in the string will be preserved, and all single-byte characters will be converted to double-byte characters.

DRDA_GRAPHIC_TO_MBCS

Property	Description
Default Value	FALSE
Range of Values	{TRUE|FALSE}
Syntax	DRDA_GRAPHIC_TO_MBCS={*TRUE*|*FALSE*}

DRDA_GRAPHIC_TO_MBCS directs the gateway to convert graphic data that has been fetched from the DRDA Server into Oracle multi-byte data, translating double-byte characters into single-byte characters where possible.

DRDA_ISOLATION_LEVEL

Property	Description
Default Value	CHG for DB2/400, CS for DB2/OS390, DB2/UDB
Range of Values	{CHG|CS|RR|ALL|NC}

Property	Description
Syntax	DRDA_ISOLATION_LEVEL={*CHG*|*CS*|*RR*|*ALL*|*NC*}

DRDA_ISOLATION_LEVEL specifies the isolation level that is defined to the package when it is created. All SQL statements that are sent to the remote DRDA database are executed with this isolation level. Isolation level seriously affects performance of applications. Use caution when specifying an isolation level other than the default. For information on isolation levels, refer to your IBM database manuals.

The following table lists the isolation levels and their descriptions. The levels are specified in ascending order of control, with CHG having the least reliable cursor stability and RR having the most. Note that higher stability uses more resources on the server and can lock those resources for extended periods.

Table B–1 Isolation Levels and Their Descriptions

Level	Description
CHG	Change (default for DB2/400)
CS	Cursor Stability (default for DB2/UDB, and DB2/OS390)
RR	Repeatable Read
ALL	ALL
NC	No Commit

Note: Any change to this parameter requires you to rebind.

DRDA_LOCAL_NODE_NAME

Property	Description
Default Value	AIX_RS6K
Range of Values	any alphanumeric string 1 to 8 characters in length
Syntax	DRDA_LOCAL_NODE_NAME=*name*

DRDA_LOCAL_NODE_NAME specifies the name by which the gateway will be known to the DRDA Server. This name is used internally by the DRDA Server to identify the local node.

DRDA_MBCS_TO_GRAPHIC

Property	Description
Default Value	FALSE
Range of Values	{TRUE|FALSE}
Syntax	DRDA_MBCS_TO_GRAPHIC={*TRUE*|*FALSE*}

DRDA_MBCS_TO_GRAPHIC directs the gateway to convert multi-byte data (that has been sent from Oracle to the DRDA database) into pure double-byte characters. This parameter is primarily intended to be used with bind variables in order to ensure that

the data is properly formatted and will therefore be acceptable to the DRDA Server. It applies only to INSERT SQL statements that are using bind variables. When used in combination with the DRDA_GRAPHIC_LIT_CHECK parameter, this parameter can help ensure that data that is being inserted into a Graphic column is handled correctly by the target DRDA Server.

DRDA_OPTIMIZE_QUERY

Property	Description
Default Value	TRUE
Range of Values	{TRUE|FALSE}
Syntax	DRDA_OPTIMIZE_QUERY={*TRUE*|*FALSE*}

DRDA_OPTIMIZE_QUERY enables or disables the distributed query optimizer (DQO) capability. The DQO capability is useful for optimizing queries that access large amount of data, but it can add overhead to small queries.

This parameter is valid only if the DRDA Server is DB2/OS390. If the DRDA Server is DB2/400 or DB2/UDB, then you must set the value to FALSE.

DRDA_PACKAGE_COLLID

Property	Description
Default Value	ORACLE
Range of Values	An alphanumeric string 1 to 18 characters in length
Syntax	DRDA_PACKAGE_COLLID=*collection_id*

DRDA_PACKAGE_COLLID specifies the package collection ID. Note that in DB2/400, the collection ID is actually the name of an AS/400 library.

Note: Any change to this parameter requires you to rebind the package.

DRDA_PACKAGE_CONSTOKEN

Property	Description
Default Value	None, use the sample provided
Range of Values	A 16-digit hexadecimal number
Syntax	DRDA_PACKAGE_CONSTOKEN=*hexnum*

DRDA_PACKAGE_CONSTOKEN specifies the package consistency token. This is a 16-digit hexadecimal representation of an 8-byte token. Oracle recommends that you do not change the consistency token. The consistency token used at runtime must match the one used when the package is bound. The value depends on the DRDA Server being used.

Note: Any change to this parameter requires you to rebind the package.

DRDA_PACKAGE_NAME

Property	Description
Default Value	G2DRSQL
Range of Values	An alphanumeric string 1 to 18 characters in length
Syntax	DRDA_PACKAGE_NAME=*name*

DRDA_PACKAGE_NAME specifies the package name. Note that the package is stored in the DRDA Server under this name as a SQL resource. Refer to the DRDA Server documentation for length limitations package names. Many typical implementations restrict the length to 8 characters.

Note: Any change to this parameter requires that you rebind the package.

DRDA_PACKAGE_OWNER

Property	Description
Default Value	None
Range of Values	Any valid user ID
Syntax	DRDA_PACKAGE_OWNER=*userid*

DRDA_PACKAGE_OWNER specifies the database user ID that owns the package. This enables the owner to be a user other than the connected user ID when the package is created. The package owner must be the same user as the owner of the ORACLE2PC table.

Note: Any change to this parameter requires you to rebind the package.

DRDA_PACKAGE_SECTIONS

Property	Description
Default Value	100
Range of Values	Any integer between 1 and 65535
Syntax	DRDA_PACKAGE_SECTIONS=*num*

DRDA_PACKAGE_SECTIONS specifies the number of cursors declared at the remote database when the package is bound. This is the maximum number of open cursors permitted at any one time. Change this parameter only if an application needs more than 100 open concurrent cursors.

Note: Any change to this parameter requires you to rebind the package.

DRDA_READ_ONLY

Property	Description
Default Value	FALSE
Range of Values	{TRUE|FALSE}
Syntax	DRDA_READ_ONLY={*TRUE*|*FALSE*}

DRDA_READ_ONLY specifies whether the gateway runs in a read-only transaction mode. In this mode, SQL statements that modify data are not permitted.

DRDA_RECOVERY_PASSWORD

Property	Description
Default Value	none
Range of Values	any valid password
Syntax	DRDA_RECOVERY_PASSWORD=*passwd*

DRDA_RECOVERY_PASSWORD is used with the DRDA_RECOVERY_USERID. The recovery user connects to the IBM database if a distributed transaction is in doubt.

DRDA_RECOVERY_USERID

Property	Description
Default Value	ORARECOV
Range of Values	Any valid user ID
Syntax	DRDA_RECOVERY_USERID=*userid*

DRDA_RECOVERY_USERID specifies the user ID that is used by the gateway if a distributed transaction becomes in doubt. This user ID must have execute privileges on the package and must be defined to the IBM database.

If a distributed transaction becomes in doubt, then the Oracle database determines the status of the transaction by connecting to the IBM database, using the DRDA_RECOVERY_USERID. If this parameter is missing, then the gateway attempts to connect to a user ID of ORARECOV.

DRDA_REMOTE_DB_NAME

Property	Description
Default Value	DB2V2R3
Range of Values	An alphanumeric string 1 to 18 characters in length

Property	Description
Syntax	DRDA_REMOTE_DB_NAME=*name*

DRDA_REMOTE_DB_NAME specifies the DRDA Server location name. This is an identifying name that is assigned to the server for DRDA purposes. A technique for determining this name by using a SQL SELECT statement is discussed in each of the server-specific installation sections in Chapter 14, "Configuring Oracle Database Gateway for DRDA" in *Oracle Database Gateway Installation and Configuration Guide for AIX 5L Based Systems (64-Bit), HP-UX PA-RISC (64-Bit), Solaris Operating System (SPARC 64-Bit), Linux x86, and Linux x86-64* and *Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows*.

FDS_CLASS

Property	Description
Default Value	TG4DRDA_DB2MVS
Range of Values	Refer to the list below for valid values
Syntax	FDS_CLASS=*TG4DRDA_DB2MVS*

FDS_CLASS specifies the capability classification used by Oracle database and the gateway. These values may change from release to release, depending on whether the gateway capabilities change.

The valid default values for FDS_CLASS are as follows:

For a DB2/OS390 database: TG4DRDA_DB2MVS

For a DB2/400 database: TG4DRDA_DB2400

For a DB2/UDB database: TG4DRDA_DB2UDB

HS_NLS_NCHAR

Property	Description
Default Value	None
Range of Values	Any valid character set specification
Syntax	HS_NLS_NCHAR=*character_set*

HS_NLS_NCHAR specifies the character set that the gateway will use to interact with the DRDA Server when accessing Graphic data. Set this parameter to the same value as the character set component of the HS_LANGUAGE parameter. For additional details, refer to Appendix C, "Globalization Support for DRDA" and to the *Oracle Database Heterogeneous Connectivity Administrator's Guide*.

LOG_DESTINATION

Property	Description
Default Value	$ORACLE_HOME/dg4drda/log/gateway sid_ pid.log

Property	Description
Range of Values	Any valid file path
Syntax	LOG_DESTINATION=*logpath*

LOG_DESTINATION specifies the destination for gateway logging and tracing. This parameter should specify a file. If the file already exists, it will be overwritten.

After any failure to open the logpath, a second attempt to open the default is made.

Usually, LOG_DESTINATION should specify a directory. If it is specified as a file, and if two or more users simultaneously use the same instance of the gateway, then they are writing to the same log. The integrity of this log is not guaranteed. If you do not specify this parameter, then the default is assumed.

ORA_MAX_DATE

Property	Description
Default Value	4712-12-31
Range of Values	Any valid date less than 4712-12-31
Syntax	ORA_MAX_DATE=*yyyy-mm-dd*

ORA_MAX_DATE specifies the gateway maximum date value. If the fetched date value is larger than 4712-12-31, the gateway replaces the date value with the value defined by the ORA_MAX_DATE parameter. Any date between January 1, 4712 BC and December 31, 4712 AD is valid.

ORA_NLS11

Property	Description
Default Value	$ORACLE_HOME/nls/data
Range of Values	Any valid Globalization Support directory path
Syntax	SET ORA_NLS11=*nlspath*

ORA_NLS11 specifies the directory to which the gateway loads its character sets and other language data. Normally this parameter does not need to be set. Some configurations, however, may require that it be set.

ORACLE_DRDA_TCTL

Property	Description
Default Value	None
Range of Values	Any valid file path
Syntax	ORACLE_DRDA_TCTL=*tracecontrolpath*

ORACLE_DRDA_TCTL specifies the path to the DRDA internal trace control file. This file contains module tracing commands. A sample file is stored in

$ORACLE_HOME/dg4drda/admin/debug.tctl. This parameter is used for diagnostic purposes.

ORACLE_DRDA_TRACE

Property	Description
Default Value	value specified for LOG_DESTINATION
Range of Values	any valid file path
Syntax	ORACLE_DRDA_TRACE=*logpath*

ORACLE_DRDA_TRACE is used to specify a different log path for DRDA internal tracing. This tracing is separate from the rest of the gateway tracing, as specified by the LOG_DESTINATION parameter. By default, this parameter will append the DRDA internal trace to the gateway trace. This parameter is used for diagnostic purposes.

TRACE_LEVEL

Property	Description
Default Value	0
Range of Values	0-255
Syntax	TRACE_LEVEL=*number*

TRACE_LEVEL specifies a code tracing level. This value determines the level of detail which is logged to the gateway logfile during execution. This parameter is primarily used for diagnostics.

HS_NLS_DATE_FORMAT

Property	Description
Default value	Value determined by the HS_LANGUAGE initialization parameter
Range of values	Any valid date format mask (up to 255 characters)

Defines the date format for dates used by the target system. This initialization parameter has the same function as the NLS_DATE_FORMAT initialization parameter for an Oracle database. The value can be any valid date mask listed in the *Oracle Database SQL Language Reference*, but must match the date format of the target system. For example, if the target system stores the date February 14, 2001 as 2001/02/14, set the parameter to yyyy/mm/dd. Note that characters must be lowercase.

HS_NLS_DATE_LANGUAGE

Property	Description
Default value	Value determined by the HS_LANGUAGE initialization parameter
Range of values	Any valid NLS_LANGUAGE value (up to 255 characters)

Specifies the language used in character date values coming from the non-Oracle system. Date formats can be language independent. For example, if the format is `dd/mm/yyyy`, all three components of the character date are numeric. In the format `dd-mon-yyyy`, however, the month component is the name abbreviated to three characters. The abbreviation is language dependent. For example, the abbreviation for the month April is "apr", which in French is "avr" (Avril).

Heterogeneous Services assumes that character date values fetched from the non-Oracle system are in this format. Also, Heterogeneous Services sends character date bind values in this format to the non-Oracle system.

HS_NLS_NUMERIC_CHARACTER

Property	Description
Default value	Value determined by the `HS_LANGUAGE` initialization parameter
Range of values	Any valid `NLS_NUMERIC_CHARACTERS` value (any two valid numeric characters)

Specifies the characters to use as the group separator and the decimal character. The group separator separates integer groups (such as thousands, millions, and billions). The decimal character separates the integer portion of a number from the decimal portion.

C

Globalization Support for DRDA

This appendix documents the Globalization Support information for the Oracle Database Gateway for DRDA. This supplements the general Globalization Support information found in the *Oracle Database Advanced Application Developer's Guide*.

Globalization Support enables users to interact with Oracle applications in their native language, using their conventions for displaying data. The Globalization Support architecture is data-driven, enabling support for specific languages and character encoding schemes to be added without any changes in source code.

There are a number of different settings in the gateway, DRDA server, Oracle database, and client that affect Globalization Support processing. In order for translations to take place correctly, character settings of these components must be compatible.

This appendix contains the following sections:

Overview of Globalization Support Interactions

Figure C–1 illustrates Globalization Support interactions within your system, including each component of your system and the parameters of each component that affect Globalization Support processing in a distributed environment. Table C–1 describes the architecture illustrated in Figure C–1.

Figure C–1 Architecture of Globalization Support Interactions with Your System Components

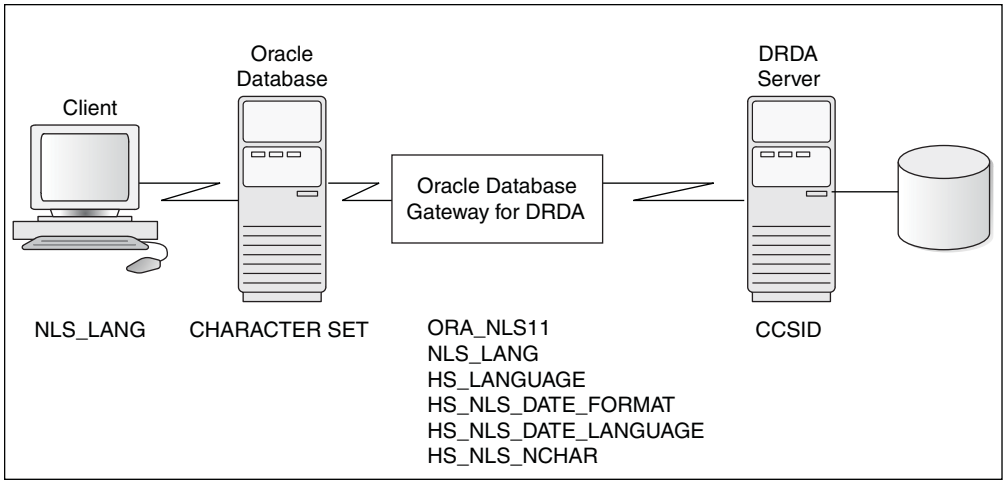


Table C–1 describes in detail the parameters and variables needed for Globalization Support processing within each of your system environments: the client environment, the Oracle database, the gateway, and the DRDA server.

Table C–1 Parameters Needed for Globalization Support Processing in Your System Environments

Environment	Parameter or Variable	Description
Client	NLS_LANG	An environmental variable. NLS_LANG sets the Globalization Support environment that is used by the database, both for the server session and for the client application. This ensures that the language environments of both database and client application are automatically the same. Because NLS_LANG is an environment variable, it is read by the client applications at startup time. The client communicates the information defined in NLS_LANG to the server when it connects. Refer to "Client and Oracle Database Configuration" on page C-4 for detailed information.
Oracle database	CHARACTER SET	This option is set during creation of the database. CHARACTER SET determines the character encoding scheme that is used by the database. CHARACTER SET is defined at database creation in the CREATE DATABASE statement. All data columns of type CHAR, VARCHAR2, and LONG have their data stored in the database character set. Refer to "Client and Oracle Database Configuration" on page C-4 for detailed information.
Oracle Database Gateway for DRDA	ORA_NLS11	An environmental variable. ORA_NLS11 determines where the gateway loads its character sets and other language data. Refer to "Gateway Language Interaction with DRDA Server" on page C-4 for detailed information.
Oracle Database Gateway for DRDA	NLS_LANG	An environmental variable. NLS_LANG defines the character set that is used for communication between the gateway and the Oracle database. Refer to "Gateway Language Interaction with DRDA Server" on page C-4 for detailed information.
Oracle Database Gateway for DRDA	HS_LANGUAGE	An initialization parameter HS_LANGUAGE defines the character set that is used for communication between the gateway and the DRDA server. Refer to "Gateway Language Interaction with DRDA Server" on page C-4 for detailed information.

Table C–1 (Cont.) Parameters Needed for Globalization Support Processing in Your System Environments

Environment	Parameter or Variable	Description
Oracle Database Gateway for DRDA	HS_NLS_NCHAR	An initialization parameter. HS_NLS_NCHAR defines the NCHAR character set that is used for communications between the gateway and the DRDA server. This parameter is required when the gateway will be accessing GRAPHIC or multibyte data on the DRDA server. Set this parameter to the same value as the character set component of the HS_LANGUAGE parameter. For detailed information, refer to "Gateway Language Interaction with DRDA Server" on page C-4.
Oracle Database Gateway for DRDA	HS_NLS_DATE_FORMAT	An initialization parameter. HS_NLS_DATE_FORMAT specifies the format for dates that are used by the DRDA server. Refer to "Gateway Language Interaction with DRDA Server" on page C-4 for detailed information.
Oracle Database Gateway for DRDA	HS_NLS_DATE_LANGUAGE	An initialization parameter. HS_NLS_DATE_LANGUAGE specifies the language that is used by the DRDA server for day and month names, and for date abbreviations. Refer to "Gateway Language Interaction with DRDA Server" on page C-4 for detailed information.
DRDA server	CCSID	CCSID is the server character set that is mapped in the gateway to the equivalent Oracle character set. The CCSID specifies the character set that the DRDA database uses to store data. It is defined when you create your database. Refer to "Gateway Codepage Map Facility" on page C-6.

Client and Oracle Database Configuration

A number of Globalization Support parameters control Globalization Support processing between the Oracle database server and client. You can set language-dependent action defaults for the server, and you can set language-dependent action for the client that overrides these defaults. For a complete description of Globalization Support parameters, refer to the Globalization Support chapter in the *Oracle Database Administrator's Guide*. These parameters do not directly affect gateway processing. However, you must ensure that the client character set (which is specified by the Oracle database NLS_LANG environment variable) is compatible with the character sets that you specify on the gateway and on the DRDA server.

When you create the Oracle database, the character set that is used to store data is specified by the CHARACTER SET clause of the CREATE DATABASE statement. After the database is created, the database character set cannot be changed unless you re-create the database.

Normally, the default for CHARACTER SET is US7ASCII, which supports only the 26 Latin alphabetic characters. If you have specified 8-bit character sets on the gateway and DRDA server, then you must have a compatible 8-bit character set defined on your database. To check the character set of an existing database, run the command:

```
SELECT USERENV('LANGUAGE') FROM DUAL;
```

For more information, refer to "Specifying Character Sets in OCI" in the *Oracle Database Globalization Support Guide*.

Note that this does not mean that the gateway character set must be the same as the Oracle database character set. The Oracle Net facility will be performing implicit conversion between the Oracle database character set and the gateway character set.

Gateway Language Interaction with DRDA Server

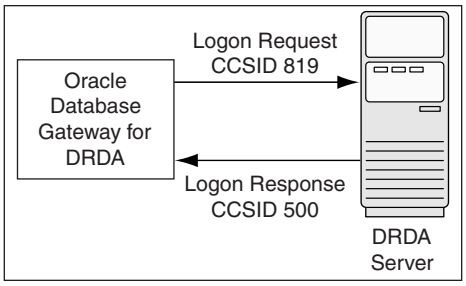
During logon of the gateway to the DRDA server, initial language information is exchanged between the Gateway and the server. First, the gateway sends to the DRDA server the CCSID it will be conversing in. In the following example, the Oracle character set "WE8ISO8859P1" is mapped to CCSID 819 (an ASCII Code Page). This CCSID is sent to the DRDA server. The DRDA server responds with the CCSID that it will be conversing in. This will be the CCSID with which the DB2 database was generated. Also, in the following example, this is CCSID 500, an EBCDIC code page. Figure C–2, "Gateway Language Interaction with DRDA Server", illustrates this process.

A DB2 instance will map unknown CCSIDs using the SYSIBM.SYSSTRINGS table (this table has different names for the various DB2 versions). It is possible to add additional character set mappings to this table using DB2 utilities. Please refer to the DB2 Installation documentation for details.

The setting of the HS_LANGUAGE parameter in the gateway init*sid*.ora determines which CCSID is used by the gateway for the conversation. Similarly, the setting of the HS_NLS_NCHAR parameter determines which CCSID will be used by the gateway for GRAPHIC data interchange. For the list of supported ASCII-based Oracle character sets that are mapped to CCSIDs, refer to "Gateway Codepage Map Facility" on page C-6.

Note again that the gateway character set need not be the same as the Oracle database character set. In many cases, it is not feasible to set the gateway character set equal to the Oracle database character set because the DRDA server will not have a valid translation for it. Instead, choose a character set that will have the most complete intersection with the character set that is used by the DRDA server. The Oracle Net facility will do any translation between the gateway character set and the Oracle database character set.

Figure C–2 Gateway Language Interaction with DRDA Server



Gateway Configuration

After the gateway is installed, you must change several parameters to customize for Globalization Support support.

Globalization Support Parameters in the Gateway Initialization File

Four parameters in the gateway initialization file (initsid.ora) affect Globalization Support:

- HS_LANGUAGE
- HS_NLS_NCHAR
- HS_NLS_DATE_FORMAT

- HS_NLS_DATE_LANGUAGE

HS_LANGUAGE

HS_LANGUAGE defines the character set that is used for communication between the gateway and the DRDA server. It specifies the conventions such as: the language used for messages from the target system; names of days and months; symbols for AD, BC, AM, and PM; and default language sorting mechanism.

The syntax of the HS_LANGUAGE parameter is:

```
HS_LANGUAGE=language[_territory.character_set]
```

where:

language can be any valid language.

territory is optional, and defaults to AMERICA.

character_set is optional and defaults to US7ASCII. This must be an ASCII base character set name, and it should match a character set listed in the gateway code page map. Refer to "Gateway Codepage Map Facility" on page C-6 for the list of supplied character set mappings.

If you omit the HS_LANGUAGE parameter from init*sid*.ora, then the default setting is AMERICAN_AMERICA.US7ASCII. EBCDIC character sets are not supported. The values for *language* and *territory* (such as AMERICAN_AMERICA) must be valid, but they have no effect on translations.

HS_NLS_NCHAR

HS_NLS_NCHAR specifies the character set that is used by the gateway to interchange GRAPHIC data. For correct compatibility, set it to the same character set name that is specified in the HS_LANGUAGE parameter. If it is set to a character set other than that specified in HS_LANGUAGE, or if it is omitted, then translation errors will occur.

HS_NLS_DATE_FORMAT

HS_NLS_DATE_FORMAT specifies the format for dates used by the DRDA server.

The syntax of the NLS_DATE_FORMAT parameter is:

```
HS_NLS_DATE_FORMAT=date_format
```

where *date_format* must be YYYY-MM-DD, the ISO date format. If this parameter is set to any other value or is omitted, then you receive an error when updating, deleting from, selecting from, or inserting into, a table with date columns.

HS_NLS_DATE_LANGUAGE

HS_NLS_DATE_LANGUAGE specifies the language used by the DRDA server for day and month names, and for date abbreviations. Because ISO date format contains numbers only, this parameter has no effect on gateway date processing and should be omitted.

Gateway Codepage Map Facility

The gateway now has a user-specifiable facility to map IBM Coded Character Set Identifiers (CCSIDs) to Oracle database character sets for data translation.

The map name defaults to `codepage.map` and is located in the directory `ORACLE_HOME/dg4drda/admin`. Refer to Appendix B, "Initialization Parameters" for more detailed information about the `DRDA_CODEPAGE_MAP` parameter.

The map has two different forms of syntax. The first form of syntax defines a mapping between a CCSID and an Oracle database character set:

```
[S|D|M] CCSID direction Oracle_CharacterSet {shift}
```

where:

`S` designates a single-byte character set

`D` designates a double-byte character set

`M` designates a multibyte character set

`CCSID` is the IBM coded character set identifier

`direction` is one of the following:

- `=` means mapping is bidirectional
- `<` means mapping is one-way, Oracle character set to CCSID
- `>` means mapping is one-way, CCSID to Oracle character set

`Oracle_CharacterSet` is the name of a valid Oracle character set.

`shift` indicates a character set that requires `Shift OUT/IN` processing. Set this attribute only for EBCDIC-based double-byte and multibyte mappings.

The second form of syntax defines a mapping of a multibyte CCSID to its single-byte and double-byte CCSID equivalents:

```
MBC multi = single double
```

where:

`multi` is the multibyte CCSID

`single` is the single-byte CCSID

`double` is the double-byte CCSID

This facility is intended as a way of mapping CCSIDs which were not previously mapped as shipped with the gateway. You must contact Oracle Support Services before modifying this map.

The following are the contents of the map as shipped with the Oracle Database Gateway for DRDA;

```
# Copyright (c) 2001, 2007, Oracle Corporation.  All rights reserved.
# Database Gateway for DRDA - CodePage/Oracle CharacterSet Map
# S==Single-byte, D==Double-byte, M==Multi-byte, MBC==SBC DBC mapping
#
# Single-byte codepage mappings
#
S    37 = WE8EBCDIC37   # United States/Canada                          EBCDIC
S   273 = D8EBCDIC273   # Austria/Germany                               EBCDIC
S   277 = DK8EBCDIC277  # Denmark/Norway                                EBCDIC
S   278 = S8EBCDIC278   # Finland/Sweden                                EBCDIC
S   280 = I8EBCDIC280   # Italy                                         EBCDIC
S   284 = WE8EBCDIC284  # Latin America/Spain                           EBCDIC
S   285 = WE8EBCDIC285  # United Kingdom                                EBCDIC
S   297 = F8EBCDIC297   # France                                        EBCDIC
#S   420 = AR8EBCDICX    # Arabic Bilingual (USA English)                EBCDIC
```

```
S   420 = AR8XBASIC     # Arabic Bilingual (USA English)                EBCDIC
S   424 = IW8EBCDIC424  # Israel (Hebrew)                               EBCDIC
S   437 = US8PC437      # Personal Computer,USA                         ASCII
S   500 = WE8EBCDIC500  # International                                 EBCDIC
S   813 = EL8ISO8859P7  # Greek                                         ASCII
S   819 = WE8ISO8859P1  # ISO/ANSI Multilingual                         ASCII
S   838 = TH8TISEBCDIC  # Thai w/Low-Tone Marks & Ancient Chars         EBCDIC
S   850 < US7ASCII      # Multilingual Page - Personal Computer         ASCII
S   850 = WE8PC850      # Multilingual Page - Personal Computer         ASCII
S   864 = AR8ISO8859P6  # Arabic - Personal Computer                    ASCII
S   870 = EE8EBCDIC870  # Latin 2, Multilingual/ROECE                   EBCDIC
S   871 = WE8EBCDIC871  # Iceland - CECP                                EBCDIC
S   875 = EL8EBCDIC875  # Greece                                        EBCDIC
S   904 > US7ASCII      # Traditional Chinese - PC-Data                 ASCII
S   912 = EE8ISO8859P2  # Latin 2 8-bit                                 ASCII
S   916 = IW8ISO8859P8  # Israel (Hebrew)                               ASCII
S  1025 = CL8EBCDIC1025 # Cyrillic, Multiling                           EBCDIC
S  1086 = IW8EBCDIC1086 # Israel                                        EBCDIC
S  1252 = WE8MSWIN1252  # Latin 1 - MS-Windows                          ASCII
S  1253 = EL8MSWIN1253  # Greek - MS-Windows                            ASCII
S 28709 > WE8EBCDIC37   # United States/Canada (CP28709==CP37)          EBCDIC
#
# Multibyte codepage mappings
#
#S   833 > KO16DBCS       # Korean Extended single-byte                 EBCDIC
#D   834 > KO16DBCS shift # Korean double-byte                          EBCDIC
#M   933 = KO16DBCS shift # Korean Mixed multi-byte                      EBCDIC

#MBC 933 = 833 834        # Korean Mixed multi-byte                     EBCDIC
#
#S  1088 > KO16MSWIN949 # Korean KS single-byte PC-Data                 ASCII
#D   951 > KO16MSWIN949 # Korean KS double-byte PC-Data                 ASCII
#M   949 = KO16MSWIN949 # Korean KS multi-byte PC-Data                  ASCII
#MBC 949 = 1088 951     # Korean KS multi-byte PC-Data                  ASCII
#
#S   891 > KO16KSC5601 # Korean single-byte                            ASCII
#S  1040 > KO16KSC5601 # Korean single-byte                            ASCII
#D   926 > KO16KSC5601 # Korean double-byte                            ASCII
#M   934 = KO16KSC5601 # Korean multi-byte                             ASCII
#M   944 > KO16KSC5601 # Korean multi-byte                             ASCII
#MBC 934 = 891 926     # Korean multi-byte                             ASCII
#MBC 944 = 1040 926    # Korean multi-byte Extended                    ASCII
#
#S 28709 > ZHT16DBCS       # Traditional Chinese single-byte           EBCDIC
#D   835 > ZHT16DBCS shift # Traditional Chinese double-byte           EBCDIC
#M   937 = ZHT16DBCS shift # Traditional Chinese multi-byte            EBCDIC
#MBC 937 = 28709 835       # Traditional Chinese multi-byte            EBCDIC
#
#S  1114 > ZHT16MSWIN950 # Traditional Chinese single-byte             ASCII
#D   947 > ZHT16MSWIN950 # Traditional Chinese double-byte             ASCII
#M   950 = ZHT16MSWIN950 # Traditional Chinese multi-byte              ASCII
#MBC 950 = 1114 947      # Traditional Chinese multi-byte              ASCII
#

#S   836 > ZHS16DBCS       # Simplified Chinese single-byte            EBCDIC
#D   837 > ZHS16DBCS shift # Simplified Chinese double-byte            EBCDIC
#M   935 = ZHS16DBCS shift # Simplified Chinese multi-byte             EBCDIC
#MBC 935 = 836 837         # Simplified Chinese multi-byte             EBCDIC
#
#S  1027 > JA16DBCS       # Japanese single-byte                       EBCDIC
```

```
#D   300 > JA16DBCS shift # Japanese double-byte                          EBCDIC
#D  4396 > JA16DBCS shift # Japanese double-byte                          EBCDIC
#M   939 = JA16DBCS shift # Japanese multi-byte                           EBCDIC
#M  5035 > JA16DBCS shift # Japanese multi-byte                           EBCDIC
#MBC  939 = 1027 300      # Japanese multi-byte                           EBCDIC
#MBC 5035 = 1027 4396     # Japanese multi-byte                           EBCDIC
#
#S   290 > JA16EBCDIC930       # Japanese single-byte                     EBCDIC
#D   300 > JA16EBCDIC930 shift # Japanese double-byte                     EBCDIC
#D  4396 > JA16EBCDIC930 shift # Japanese double-byte                     EBCDIC
#M   930 = JA16EBCDIC930 shift # Japanese multi-byte                      EBCDIC
#M  5026 > JA16EBCDIC930 shift # Japanese multi-byte                      EBCDIC
#MBC  930 = 290 300            # Japanese multi-byte                      EBCDIC
#MBC 5026 = 290 4396           # Japanese multi-byte                      EBCDIC
#
```

Refer to the following list to check the character set of an existing database:

- **for DB2/OS390:** Ask your system administrator. There is no single command you use.
- **for DB2/400:** Run the command `DSPSYSVAL SYSVAL(QCCSID)`
- **for DB2/UDB:** Ask your system administrator. There is no single command you use.

Multibyte and Double-Byte Support in the Gateway

To enable the gateway to properly handle double-byte and multibyte data, you must configure the code page map facility with proper multibyte maps and (as an option) you can set the following gateway configuration parameters:

- `DRDA_GRAPHIC_LIT_CHECK`
- `DRDA_GRAPHIC_TO_MBCS`
- `DRDA_MBCS_TO_GRAPHIC`
- `DRDA_GRAPHIC_PAD_SIZE`
- `DRDA_GRAPHIC_CHAR_SIZE`

Refer to Appendix B, "Initialization Parameters", for the values of these parameters.

Configuring the code page map requires knowledge of the code pages that have been configured in the DRDA server database as well as knowledge of compatible Oracle database character sets.

IBM coded character set identifiers (CCSIDs) are used to indicate which code pages are configured as the primary codepage for the database, as well as any translation character sets loaded into the database. Some DRDA servers, such as with DB2, have a translation facility in which character set transforms are mapped between two compatible character sets. For DB2/OS390, these transforms are stored in the table `SYSIBM.SYSSTRINGS` and transform on the CCSID codepage to another CCSID codepage. In SYSSTRINGS, IN and OUT columns specify the CCSIDs that are used in the transform. Typical transforms are from ASCII to EBCDIC and back again. Two transforms are therefore used for two given CCSIDs.

Multibyte codepages are a composite of a single-byte codepage and a double-byte codepage. As an example, the Korean EBCDIC multi-byte codepage, CCSID 933, is composed of two codepages, codepage 833 (for single-byte) and codepage 834 (for double-byte). The DRDA server, therefore, can send data to the gateway in any of

these three codepages, and the gateway must translate suitably depending on which codepage the data is associated with. Because CCSID 933 is an EBCDIC-based codepage, and the gateway must use an ASCII-based codepage, we identify an equivalent set of codepages, which are ASCII-based. An example would be the Korean multibyte codepage, CCSID 949, which is composed of two codepages, codepage 1088 (for single-byte) and codepage 951 (for double-byte).

The codepage map facility is used to map these CCSIDs into the equivalent Oracle database character sets. Unlike IBM CCSIDs, Oracle database character sets are unified (in that single-byte and double-byte character sets have been combined into one set) and are thus identified by one ID instead of three IDs. In our previous example, the equivalent Oracle database character set for the ASCII Korean codepages would be `KO16MSWIN949`, and the EBCDIC Korean codepages would be KO16DBCS. These are identified to the gateway by using a set of mapping entries in the `codepage.map` file.

First, the EBCDIC Korean sets are:

```
S   833 > KO16DBCS       # Korean Extended single-byte                EBCDIC
D   834 > KO16DBCS shift # Korean double-byte                         EBCDIC
M   933 = KO16DBCS shift # Korean Mixed multi-byte                    EBCDIC
MBC 933 = 833 834        # Korean Mixed multi-byte                    EBCDIC
```

Notice that the multibyte set is a bidirectional map to `KO16DBCS`, while the single and double codepages are mapped one-way to `KO16DBCS`. Because only one bidirectional CCSID to Oracle database character set entry for a given pair can exist, we directly map the multibyte sets. And because the single-byte and double-byte CCSIDs are ostensibly subsets of `KO16DBCS`, we map them as one-way entries. Note that double-byte and multibyte maps are tagged with the shift attribute. This is required for EBCDIC double-byte and multibyte codepages as part of the shift out/in encapsulation of data. Note that the single-byte map is not marked because single-byte sets are not permitted to contain double-byte data and thus will never use shift encapsulation. Also note that the MBC entry ties the codepages together.

The ASCII Korean sets are similarly mapped and are:

```
S  1088 > KO16MSWIN949 # Korean KS single-byte PC-Data               ASCII
D   951 > KO16MSWIN949 # Korean KS double-byte PC-Data               ASCII
M   949 = KO16MSWIN949 # Korean KS multi-byte PC-Data                ASCII
MBC 949 = 1088 951     # Korean KS multi-byte PC-Data                ASCII
```

Notice that the multibyte set is a bidirectional map to `KO16MSWIN949`, while the single and double codepages are mapped one-way to `KO16MSWIN949`. Because only one bidirectional CCSID to Oracle database character set entry for a given pair can exist, we directly map the multibyte sets. And because the single-byte and double-byte CCSIDs are ostensibly subsets of `KO16MSWIN949`, we map them as one-way entries. Note that there is no shift attribute in any of these mappings. This is because ASCII-based sets do not use shift out/in encapsulation. Instead, ASCII-based sets use a different method (which does not use a shift out/in protocol) to identify double-byte characters.

The preceding entries supply the necessary codepage mappings for the gateway. To complete the example, we need to specify the correct character set in the `HS_LANGUAGE` and `HS_NLS_NCHAR` parameters in the gateway initialization file. The gateway initialization parameters would look as follows:

```
HS_LANGUAGE=AMERICAN_AMERICA.KO16MSWIN949
HS_NLS_NCHAR=KO16MSWIN949
```

Note that the specified character set must be ASCII-based.

This takes care of configuration of the gateway. The last step is to set up transforms between the EBCDIC codepages and the ASCII codepages in the DRDA server database. Normally, the gateway would use a total of six transforms, one of each pair in both directions. You may save some table space by installing only the ASCII-to-EBCDIC transforms. The reasoning is that the DRDA server needs to translate only the ASCII data that is sent by the gateway, but the DRDA server does not need to send ASCII data. The gateway will receive the EBCDIC data and translate as needed. This one-sided data transfer methodology is called "receiver-makes-right", meaning that the receiver must translate whatever character set the sender uses. In our example, the DRDA server is EBCDIC-based, so it will send all data in EBCDIC. The server, therefore, does not need to have an EBCDIC-to-ASCII transform because the server will never use the transform.

In our previous example, the DRDA server database is assumed to be EBCDIC, which is likely to be true for a DB2/OS390 database. For a DB2/UDB database, however, this is not likely to be true. Because most DB2/UDB databases are running on ASCII-based computers, they will likely be created with ASCII-based codepages. In such cases, the gateway needs to have only one set of codepage map definitions, which are those for the ASCII set. Also, because both the DRDA server and the gateway will be using the same codepages, no character set transforms need to be loaded into the DB2 database. This can help reduce the amount of CPU overhead that is associated with character translation.

One final note concerning codepage map entries: Be aware that some multi-byte codepages may be composed of single-byte CCSIDs that are already defined in the codepage.map file that is provided with the product. If you are adding a new set of entries to support a multibyte set, then comment out the provided entries so that your new entries will be used correctly.

Additional codepage mappings, which are not already provided, are possible. You may construct entries such as those in our examples, given knowledge of the IBM CCSIDs and the Oracle database character sets. Because this can be complex (given the IBM documentation of codepage definitions and Oracle database character set definitions), thoroughly test your definitions for all desired character data values before putting them into production.

If you are uncertain, then contact Oracle Support Services to request proper codepage mapping entries.

Message Availability

Whether a language message module is available depends on which modules are installed in the Oracle product set running on the server. If message modules for a particular language set are not installed, then specifying that language with a language parameter does not display messages in the requested language.

Example of Globalization Support Configuration

Following is an example of all the settings needed to configure the gateway, DRDA server, Oracle database, and client so that a language and character set are working compatibly across the system. In this example, the settings enable a customer in Germany to interact with the gateway in German:

Gateway init*sid*.ora file:

```
HS_LANGUAGE=AMERICAN_AMERICA.WE8ISO8859P1
HS_NLS_DATE_FORMAT=YYYY-MM-DD
```

DRDA Server CCSID:

```
273 (D8EBCDIC273)
```

Oracle Database and client setting for database:

```
SELECT USERENV('language') FROM DUAL;
USERENV('LANGUAGE')
------------------------------
AMERICAN_AMERICA.WE8ISO8859P1
```

Oracle Database and client environment variables:

```
NLS_LANG=GERMAN_GERMANY.WE8ISO8859P1
```

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z