Oracle® Database Gateway for SQL Server

User's Guide,

11*g* Release 1 (11.1)

B31049-03

September 2007

ORACLE®

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

B31049-03

Primary Author: Maitreyee Chaliha

Contributor: Vira Goorah, Juan Pablo Ahues-Vasquez, Peter Castro, Charles Benet, Peter Wong, and Govind Lakkoju

Contents

C Data Dictionary

D Initialization Parameters

Index

List of Tables

Preface

This manual describes the Oracle Database Gateway for SQL Server, which enables Oracle client applications to access SQL Server data through Structured Query Language (SQL). The gateway, with the Oracle database, creates the appearance that all data resides on a local Oracle database, even though the data can be widely distributed.

This preface covers the following topics:

Audience

This manual is intended for Oracle database administrators who perform the following tasks:

- Installing and configuring the Oracle Database Gateway for SQL Server
- Diagnosing gateway errors
- Using the gateway to access SQL Server data

> **Note:** You should understand the fundamentals of Oracle Database Gateways and the Microsoft Windows operating system before using this guide to install or administer the gateway.

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, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documentation

For more information, see the following documents:

- *Oracle Database New Features Guide*
- *Oracle Call Interface Programmer's Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database Advanced Application Developer's Guide*
- *Oracle Database Concepts*
- *Oracle Database Performance Tuning Guide*
- *Oracle Database Error Messages*
- *Oracle Database Globalization Support Guide*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Database Net Services Administrator's Guide*
- *SQL*Plus User's Guide and Reference*
- *Oracle Database Heterogeneous Connectivity Administrator's Guide*
- *Oracle Database 2 Day DBA*
- *Oracle Database Security Guide*

Many of the examples in this book use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

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

`http://otn.oracle.com/technology/membership`

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

`http://otn.oracle.com/technology/documentation/`

Conventions

The following text conventions are used in this document:

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

1 Introduction

This chapter introduces the challenge faced by organizations when running several different database systems. It briefly covers Heterogeneous Services, the technology that the Oracle Database Gateway for SQL Server is based on.

To get a good understanding of generic gateway technology, Heterogeneous Services, and how Oracle Database Gateways fit in the picture, reading *Oracle Database Heterogeneous Connectivity Administrator's Guide* first is highly recommended.

This chapter contains the following sections:

Overview

Heterogeneous data access is a problem that affects a lot of companies. A lot of companies run several different database systems. Each of these systems stores data and has a set of applications that run against it. Consolidation of this data in one database system is often hard-in large part because many of the applications that run against one database may not have an equivalent that runs against another. Until such time as migration to one consolidated database system is made feasible, it is necessary for the various heterogeneous database systems to interoperate.

Oracle Database Gateways provide the ability to transparently access data residing in a non-Oracle system from an Oracle environment. This transparency eliminates the need for application developers to customize their applications to access data from different non-Oracle systems, thus decreasing development efforts and increasing the mobility of the application. Applications can be developed using a consistent Oracle interface for both Oracle and SQL Server.

Gateway technology is composed of two parts: a component that has the generic technology to connect to a non-Oracle system, which is common to all the non-Oracle systems, called Heterogeneous Services, and a component that is specific to the non-Oracle system that the gateway connects to. Heterogeneous Services, in conjunction with the Oracle Database Gateway agent, enables transparent access to non-Oracle systems from an Oracle environment.

Heterogeneous Services Technology

Heterogeneous Services provides the generic technology for connecting to non-Oracle systems. As an integrated component of the database, Heterogeneous Services can

exploit features of the database, such as the powerful SQL parsing and distributed optimization capabilities.

Heterogeneous Services extend the Oracle SQL engine to recognize the SQL and procedural capabilities of the remote non-Oracle system and the mappings required to obtain necessary data dictionary information. Heterogeneous Services provides two types of translations: the ability to translate Oracle SQL into the proper dialect of the non-Oracle system as well as data dictionary translations which displays the metadata of the non-Oracle system in the local format. For situations where no translations are available, native SQL can be issued to the non-Oracle system using the pass-through feature of Heterogeneous Services.

Heterogeneous Services also maintains the transaction coordination between Oracle and the remote non-Oracle system, such as providing the two-phase commit protocol to ensure distributed transaction integrity, even for non-Oracle systems that do not natively support two-phase commit.

> **See Also:** *Oracle Database Heterogeneous Connectivity Administrator's Guide* for more information about Heterogeneous Services.

Oracle Database Gateways

The capabilities, SQL mappings, data type conversions, and interface to the remote non-Oracle system are contained in the gateway. The gateway interacts with Heterogeneous Services to provide the transparent connectivity between Oracle and non-Oracle systems.

The gateway can be installed on any machine independent of the Oracle or non-Oracle database. It can be the same machine as the Oracle database or on the same machine as the SQL Server database or on a third machine as a standalone.

2

SQL Server Gateway Features and Restriction

After the gateway is installed and configured, you can use the gateway to access SQL Server data, pass SQL Server commands from applications to the SQL Server database, perform distributed queries, and copy data.

This chapter contains the following sections:

Using the Pass-Through Feature

The gateway can pass SQL Server commands or statements from the application to the SQL Server database using the DBMS_HS_PASSTHROUGH package.

Use the DBMS_HS_PASSTHROUGH package in a PL/SQL block to specify the statement to be passed to the SQL Server database, as follows:

```
DECLARE
    num_rows INTEGER;
BEGIN
    num_rows := DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@MSQL('command');
END;
/
```

Where *command* cannot be one of the following:

- BEGIN TRANSACTION
- COMMIT
- ROLLBACK
- SAVE
- SHUTDOWN
- RELEASE
- SAVEPOINT
- CONNECT

- SQL Server tool commands

The DBMS_HS_PASSTHROUGH package supports passing bind values and executing SELECT statements.

Note: TRUNCATE cannot be used in a pass-through statement.

See Also: *Oracle Database PL/SQL Packages and Types Reference* and Chapter 3, Features of Oracle Database Gateways, of *Oracle Database Heterogeneous Connectivity Administrator's Guide* for more information about the DBMS_HS_PASSTHROUGH package.

Executing Stored Procedures and Functions

Using the procedural feature, the gateway can execute stored procedures that are defined in the SQL Server database. It is not necessary to relink the gateway or define the procedure to the gateway, but the procedure's access privileges must permit access by the user that the gateway is logging in as.

See Also: *Oracle Database Heterogeneous Connectivity Administrator's Guide* for more information about executing stored procedures.

Standard PL/SQL statements are used to execute a stored procedure.

The gateway supports stored procedures in three mutually exclusive modes:

- Normal mode: Have access to IN/OUT arguments only
- Return value mode: Have a return value for all stored procedures
- Resultset mode: Out values are available as last result set

Remote User-defined Function Support

User-defined functions in a remote non-Oracle database can be used in SQL statements.

See Also: *Oracle Database Heterogeneous Connectivity Administrator's Guide* for more information about executing user-defined functions on a non-Oracle database.

Return Values and Stored Procedures

By default, all stored procedures and functions do not return a return value to the user. To enable return values, set the HS_FDS_PROC_IS_FUNC parameter in the initialization parameter file.

See Also: Appendix D, "Initialization Parameters" for information about both editing the initialization parameter file and the HS_FDS_PROC_IS_FUNC parameter.

Note: If you set the HS_FDS_PROC_IS_FUNC gateway initialization parameter, you must change the syntax of the procedure execute statement for all existing stored procedures.

In the following example, the employee name JOHN SMYTHE is passed to the SQL Server stored procedure REVISE_SALARY. The stored procedure retrieves the salary value from the SQL Server database to calculate a new yearly salary for JOHN SMYTHE. The revised salary returned in RESULT is used to update EMP in a table of an Oracle database:

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

The procedural feature automatically converts non-Oracle data types to and from PL/SQL data types.

Result Sets and Stored Procedures

The Oracle Database Gateway for SQL Server provides support for stored procedures which return result sets.

By default, all stored procedures and functions do not return a result set to the user. To enable result sets, set the HS_FDS_RESULTSET_SUPPORT parameter in the initialization parameter file.

See Also: Appendix D, "Initialization Parameters" for information about both editing the initialization parameter file and the HS_FDS_RESULTSET_SUPPORT parameter. For further information about Oracle support for result sets in non-Oracle databases see *Oracle Database Heterogeneous Connectivity Administrator's Guide*.

Note: If you set the HS_FDS_RESULTSET_SUPPORT gateway initialization parameter, you must change the syntax of the procedure execute statement for all existing stored procedures or errors will occur.

When accessing stored procedures with result sets through the Oracle Database Gateway for SQL Server, you will be in the sequential mode of Heterogeneous Services.

The Oracle Database Gateway for SQL Server returns the following information to Heterogeneous Services during procedure description:

- All the input arguments of the remote stored procedure

- None of the output arguments
- One out argument of type ref cursor (corresponding to the first result set returned by the stored procedure)

Client programs have to use the virtual package function `dbms_hs_result_set.get_next_result_set` to get the ref cursor for subsequent result sets. The last result set returned is the out argument from the procedure.

The limitations of accessing result sets are the following:

- Result sets returned by a remote stored procedure have to be retrieved in the order in which they were placed on the wire
- On execution of a stored procedure, all result sets returned by a previously executed stored procedure will be closed (regardless of whether the data has been completely retrieved or not)

In the following example, the SQL Server stored procedure is executed to fetch the contents of the `emp` and `dept` tables from SQL Server:

```
create procedure REFCURPROC (@arg1 varchar(255), @arg2 varchar(255) output)
as
select @arg2 = @arg1
select * from EMP
select * from DEPT
go
```

This stored procedure assigns the input parameter arg1 to the output parameter arg2, opens the query `SELECT * FROM EMP` in ref cursor rc1, and opens the query `SELECT * FROM DEPT` in ref cursor rc2.

OCI Program Fetching from Result Sets in Sequential Mode

The following example shows OCI program fetching from result sets in sequential mode:

```
OCIEnv *ENVH;
OCISvcCtx *SVCH;
OCIStmt *STMH;
OCIError *ERRH;
OCIBind *BNDH[3];
OraText arg1[20];
OraText arg2[255];
OCIResult *rset;
OCIStmt *rstmt;
ub2 rcode[3];
ub2 rlens[3];
sb2 inds[3];
OraText *stmt = (OraText *) "begin refcurproc@MSQL(:1,:2,:3); end;";
OraText *n_rs_stm = (OraText *)
  "begin :ret := DBMS_HS_RESULT_SET.GET_NEXT_RESULT_SET@MSQL; end;";

/* Prepare procedure call statement */

/* Handle Initialization code skipped */
OCIStmtPrepare(STMH, ERRH, stmt, strlen(stmt), OCI_NTV_SYNTAX, OCI_DEFAULT);

/* Bind procedure arguments */
inds[0] = 0;
strcpy((char *) arg1, "Hello World");
rlens[0] = strlen(arg1);
```

```
OCIBindByPos(STMH, &BNDH[0], ERRH, 1, (dvoid *) arg1, 20, SQLT_CHR,
             (dvoid *) &(inds[0]), &(rlens[0]), &(rcode[0]), 0, (ub4 *) 0,
             OCI_DEFAULT);

inds[1] = -1;
OCIBindByPos(STMH, &BNDH[1], ERRH, 1, (dvoid *) arg2, 20, SQLT_CHR,
             (dvoid *) &(inds[1]), &(rlens[1]), &(rcode[1]), 0, (ub4 *) 0,
             OCI_DEFAULT);

inds[2] = 0;
rlens[2] = 0;
OCIDescriptorAlloc(ENVH, (dvoid **) &rset, OCI_DTYPE_RSET, 0, (dvoid **) 0);
OCIBindByPos(STMH, &BNDH[2], ERRH, 2, (dvoid *) rset, 0, SQLT_RSET,
             (dvoid *) &(inds[2]), &(rlens[2]), &(rcode[2]),
             0, (ub4 *) 0, OCI_DEFAULT);

/* Execute procedure */
OCIStmtExecute(SVCH, STMH, ERRH, 1, 0, (CONST OCISnapshot *) 0,
               (OCISnapshot *) 0, OCI_DEFAULT);

/* Convert result set to statement handle */
OCIResultSetToStmt(rset, ERRH);
rstmt = (OCIStmt *) rset;

/* After this the user can fetch from rstmt */
/* Issue get_next_result_set call to get handle to next_result set */
/* Prepare Get next result set procedure call */

OCIStmtPrepare(STMH, ERRH, n_rs_stm, strlen(n_rs_stm), OCI_NTV_SYNTAX,
               OCI_DEFAULT);

/* Bind return value */
OCIBindByPos(STMH, &BNDH[1], ERRH, 1, (dvoid *) rset, 0, SQLT_RSET,
             (dvoid *) &(inds[1]), &(rlens[1]), &(rcode[1]),
             0, (ub4 *) 0, OCI_DEFAULT);

/* Execute statement to get next result set*/
OCIStmtExecute(SVCH, STMH, ERRH, 1, 0, (CONST OCISnapshot *) 0,
               (OCISnapshot *) 0, OCI_DEFAULT);

/* Convert next result set to statement handle */
OCIResultSetToStmt(rset, ERRH);
rstmt = (OCIStmt *) rset;

/* Now rstmt will point to the second result set returned by the
remote stored procedure */

/* Repeat execution of get_next_result_set to get the output arguments */
```

PL/SQL Program Fetching from Result Sets in Sequential Mode

Assume that the table `loc_emp` is a local table exactly like the SQL Server `emp` table. The same assumption applies for `loc_dept`. `outargs` is a table with columns corresponding to the out arguments of the SQL Server stored procedure.

```
create or replace package rcpackage is
  type RCTYPE is ref cursor;
end rcpackage;
/
declare
  rc1 rcpackage.rctype;
```

```
  rec1 loc_emp%rowtype;
  rc2 rcpackage.rctype;
  rec2 loc_dept%rowtype;
  rc3 rcpackage.rctype;
  rec3 outargs%rowtype;
  out_arg varchar2(255);

begin

  -- Execute procedure
  out_arg := null;
  refcurproc@MSQL('Hello World', out_arg, rc1);

  -- Fetch 20 rows from the remote emp table and insert them into loc_emp
  for i in 1 .. 20 loop
    fetch rc1 into rec1;
    insert into loc_emp (rec1.empno, rec1.ename, rec1.job,
    rec1.mgr, rec1.hiredate, rec1.sal, rec1.comm, rec1.deptno);
  end loop;

  -- Close ref cursor
  close rc1;

  -- Get the next result set returned by the stored procedure
  rc2 := dbms_hs_result_set.get_next_result_set@MSQL;

  -- Fetch 5 rows from the remote dept table and insert them into loc_dept
  for i in 1 .. 5 loop
    fetch rc2 into rec2;
    insert into loc_dept values (rec2.deptno, rec2.dname, rec2.loc);
  end loop;

  --Close ref cursor
  close rc2;

  -- Get the output arguments from the remote stored procedure
  -- Since we are in sequential mode, they will be returned in the
  -- form of a result set
  rc3 := dbms_hs_result_set.get_next_result_set@MSQL;

  -- Fetch them and insert them into the outarguments table
  fetch rc3 into rec3;
  insert into outargs (rec3.outarg, rec3.retval);

  -- Close ref cursor
  close rc3;

end;
/
```

Database Compatibility Issues for SQL Server

SQL Server and Oracle databases function differently in some areas, causing compatibility problems. The following compatibility issues are described in this section:

- Implicit Transactions (Chained Mode)
- Column Definitions

Implicit Transactions (Chained Mode)

The gateway supports the ANSI-standard implicit transactions. SQL Server stored procedures must be written for this mode. Running implicit transactions allows the gateway to extend the Oracle two-phase commit protection to transactions updating Oracle and SQL Server databases.

Column Definitions

By default, a SQL Server table column cannot contain null values unless `NULL` is specified in the column definition. SQL Server assumes all columns cannot contain null values unless you set a SQL Server option to override this default.

For an Oracle table, null values are allowed in a column unless `NOT NULL` is specified in the column definition.

Naming Rules

Naming rule issues include the following:

Rules for Naming Objects

Oracle and SQL Server use different database object naming rules. For example, the maximum number of characters allowed for each object name can be different. Also, the use of single and double quotation marks, case sensitivity, and the use of alphanumeric characters can all be different.

> **See Also:** *Oracle Database Reference* and SQL Server documentation.

Case Sensitivity

The Oracle database defaults to uppercase unless you surround identifiers with double quote characters. For example, to refer to the SQL Server table called `emp`, enter the name with double quote characters, as follows:

```
SQL> SELECT * FROM "emp"@MSQL;
```

However, to refer to the SQL Server table called `emp` owned by Scott from an Oracle application, enter the following:

```
SQL> SELECT * FROM "Scott"."emp"@MSQL;
```

If the SQL Server table called `emp` is owned by `SCOTT`, a table owner name in uppercase letters, you can enter the owner name without double quote characters, as follows:

```
SQL> SELECT * FROM SCOTT."emp"@MSQL;
```

Or

```
SQL> SELECT * FROM scott."emp"@MSQL;
```

Oracle recommends that you surround all SQL Server object names with double quote characters and use the exact letter case for the object names as they appear in the SQL Server data dictionary. This convention is not required when referring to the supported Oracle data dictionary tables or views listed in Appendix C, "Data Dictionary".

If existing applications cannot be changed according to these conventions, create views in Oracle to associate SQL Server names to the correct letter case. For example, to refer to the SQL Server table `emp` from an existing Oracle application by using only uppercase names, define the following view:

```
SQL> CREATE VIEW EMP (EMPNO, ENAME, SAL, HIREDATE)
      AS SELECT "empno", "ename", "sal", "hiredate"
      FROM "emp"@MSQL;
```

With this view, the application can issue statements such as the following:

```
SQL> SELECT EMPNO, ENAME FROM EMP;
```

Using views is a workaround solution that duplicates data dictionary information originating in the SQL Server data dictionary. You must be prepared to update the Oracle view definitions whenever the data definitions for the corresponding tables are changed in the SQL Server database.

Data Types

Data type issues include the following:

- Binary Literal Notation
- Bind Variables With LONG Columns
- Data Type Conversion

Binary Literal Notation

Oracle SQL uses hexadecimal digits surrounded by single quotes to express literal values being compared or inserted into columns defined as data type `RAW`.

This notation is not converted to syntax compatible with the SQL Server `VARBINARY` and `BINARY` data types (a `0x` followed by hexadecimal digits, surrounded by single quotes).

For example, the following statement is not supported:

```
SQL> INSERT INTO BINARY_TAB@MSQL VALUES ('0xff')
```

Where `BINARY_TAB` contains a column of data type `VARBINARY` or `BINARY`. Use bind variables when inserting into or updating `VARBINARY` and `BINARY` data types.

Bind Variables With LONG Columns

The gateway does not support using bind variables to update columns of data type `LONG`.

Data Type Conversion

SQL Server does not support implicit date conversions. Such conversions must be explicit.

For example, the gateway issues an error for the following SELECT statement:

```
SELECT DATE_COL FROM TEST@
MSQL
 WHERE DATE_COL = "1-JAN-2004";
```

To avoid problems with implicit conversions, add explicit conversions, as in the following:

```
SELECT DATE_COL FROM TEST@
MSQL
 WHERE DATE_COL = TO_DATE("1-JAN-2004")
```

> **See Also:** Appendix A, "Data Type Conversion" for more information about restrictions on data types.

Queries

Query issues include the following:

- Row Selection
- Empty Strings
- Empty Bind Variables

Row Selection

SQL Server evaluates a query condition for all selected rows before returning any of the rows. If there is an error in the evaluation process for one or more rows, no rows are returned even though the remaining rows satisfy the condition.

Oracle evaluates the query condition row-by-row and returns a row when the evaluation is successful. Rows are returned until a row fails the evaluation.

Empty Strings

Oracle processes an empty string in a SQL statement as a null value. SQL Server processes an empty string as an empty string.

When comparing an empty string the gateway passes literal empty strings to the SQL Server database without any conversion. If you intended an empty string to represent a null value, SQL Server does not process the statement that way; it uses the empty string.

You can avoid this problem by using NULL or IS NULL in the SQL statement instead of the empty string syntax, as in the following example:

```
SELECT * from "emp"@MSQL where "ename" IS NULL;
```

Selecting an empty string

For VARCHAR columns, the gateway returns an empty string to the Oracle database as NULL value.

For CHAR columns, the gateway returns the full size of the column with each character as empty space (' ').

Empty Bind Variables

For VARCHAR bind variables, the gateway passes empty bind variables to the SQL Server database as a NULL value.

Locking

The locking model for an SQL Server database differs significantly from the Oracle model. The gateway depends on the underlying SQL Server behavior, so Oracle applications that access SQL Server through the gateway can be affected by the following possible scenarios:

- Read access might block write access
- Write access might block read access
- Statement-level read consistency is not guaranteed

> **See Also:** SQL Server documentation for information about the SQL Server locking model.

Known Restrictions

If you encounter incompatibility problems not listed in this section or in "Known Problems" on page 2-14, contact Oracle Support Services. The following section describes the known restrictions and includes suggestions for dealing with them when possible:

Note: If you have any questions or concerns about the restrictions, contact Oracle Support Services.

Multiple Open Statements

Accessing SQL Server has the limitation that one open statement or cursor is allowed for each connection. If a second statement or cursor needs to open in the same transaction to access SQL Server, it requires a new connection.

Because of this limitation multiple open statements or cursors within the same transaction can lock each other because they use different connections to SQL Server.

To avoid this restriction, issue a commit, or modify the logic, or both.

Transactional Integrity

The gateway cannot guarantee transactional integrity in the following cases:

- When a statement that is processed by the gateway causes an implicit commit in the target database
- When the target database is configured to work in autocommit mode

> **Note:** Oracle strongly recommends the following:
>
> - If you know that executing a particular statement causes an implicit commit in the target database, then ensure that this statement is executed in its own transaction.

The gateway sets Autocommit Mode to Off when a connection is established to the SQL Server database.

Transaction Capability

The gateway does not support savepoints. If a distributed update transaction is under way involving the gateway, and a user attempts to create a savepoint, the following error occurs:

```
ORA-02070: database dblink does not support savepoint in this context
```

By default, the gateway is configured as `COMMIT_CONFIRM`.

COMMIT or ROLLBACK in PL/SQL Cursor Loops Closes Open Cursors

Any `COMMIT` or `ROLLBACK` issued in a PL/SQL cursor loop closes all open cursors, which can result in the following error:

```
ORA-1002:  fetch out of sequence
```

To prevent this error, move the `COMMIT` or `ROLLBACK` statement outside the cursor loop.

Stored Procedures

Changes issued through stored procedures that embed commits or rollbacks cannot be controlled by the Oracle transaction manager or Oracle `COMMIT` or `ROLLBACK` commands.

When accessing stored procedures with result sets through the Oracle Database Gateway for SQL Server, you must work in the sequential mode of Heterogeneous Services.

When accessing stored procedures with multiple result sets through the Oracle Database Gateway for SQL Server, you must read all the result sets before continuing.

Output parameters of stored procedures must be initialized to an empty string.

Pass-Through Feature

If the SQL statements being passed through the gateway result in an implicit commit at the SQL Server database, the Oracle transaction manager is unaware of the commit and an Oracle `ROLLBACK` command cannot be used to roll back the transaction.

DDL Statements

SQL Server requires some DDL statements to be executed in their own transaction, and only one DDL statement can be executed in a given transaction.

If you use these DDL statements in a SQL Server stored procedure and you execute the stored procedure through the gateway using the procedural feature, or, if you execute the DDL statements through the gateway using the pass-through feature, an error condition might result. This is because the procedural feature and the pass-through feature of the gateway cannot guarantee that the DDL statements are executed in their own separate transaction.

The following SQL Server DDL statements can cause an error condition if you attempt to pass them with the gateway pass-through feature, or if you execute a SQL Server stored procedure that contains them:

Table 2–1 Restricted DDL Statements

Statement Name
ALTER DATABASE
CREATE DATABASE
CREATE INDEX
CREATE PROCEDURE
CREATE TABLE
CREATE VIEW
DISK INIT
DROP <*object*>
DUMP TRANSACTION
GRANT
LOAD DATABASE
LOAD TRANSACTION
RECONFIGURE
REVOKE
SELECT INTO
TRUNCATE TABLE
UPDATE STATISTICS

See Also: SQL Server documentation for more information about DDL statements.

SQL Syntax

This section lists restrictions on the following SQL syntax:

- WHERE CURRENT OF Clause
- CONNECT BY Clause
- Functions in Subqueries

- Parameters in Subqueries
- Data Dictionary Table and Views in UPDATE Statement
- ROWID
- TO_DATE
- EXPLAIN PLAN Statement
- Callback Support

> **See Also:** Appendix B, "Supported SQL Syntax and Functions" for more information about restrictions on SQL syntax.

WHERE CURRENT OF Clause

`UPDATE` and `DELETE` statements with the `WHERE CURRENT OF` clause are not supported by the gateway because they rely on the Oracle `ROWID` implementation. To update or delete a specific row through the gateway, a condition style `WHERE` clause must be used.

CONNECT BY Clause

The gateway does not support the `CONNECT BY` clause in a `SELECT` statement.

Functions in Subqueries

Bind variables and expressions are not supported as operands in string functions or mathematical functions, when part of subquery in an `INSERT`, `UPDATE`, or `DELETE` SQL statement.

Parameters in Subqueries

Due to a limitation in SQL Server, you cannot use parameters in subqueries.

Data Dictionary Table and Views in UPDATE Statement

Data dictionary tables and views in the `SET` clause of an `UPDATE` statement are not supported.

ROWID

The Oracle `ROWID` implementation is not supported.

TO_DATE

`TO_DATE` is a reserved word and cannot be used as a database identifier name.

EXPLAIN PLAN Statement

The `EXPLAIN PLAN` statement is not supported.

Callback Support

SQL statements that require the gateway to callback to Oracle database would not be supported.

The following categories of SQL statements will result in a callback:

- Any DML with a sub-select, which refers to a table in Oracle database. For example:

```
INSERT INTO emp@non_oracle SELECT * FROM oracle_emp;
```

- Any DELETE, INSERT, UPDATE or "SELECT... FOR UPDATE..." SQL statement containing SQL functions or statements that need to be executed at the originating Oracle database.

  These SQL functions include USER, USERENV, and SYSDATE, and the SQL statements are in selects of data from the originating Oracle database. For example:

  ```
  DELETE FROM emp@non_oracle WHERE hiredate > SYSDATE;

  SELECT ename FROM tkhoemp@non_oracle
  WHERE hiredate IN (SELECT hiredate FROM tkhoemp)
  FOR UPDATE OF empno;
  ```

- Any SQL statement that involves a table in Oracle database, and a LONG or LOB column in a remote table. For example:

  ```
  SELECT a.long1, b.empno FROM scott.table@non_oracle a, emp b
  WHERE a.id=b.empno;

  SELECT a.long1, b.dummy FROM table_non@non_oracle a, dual b;
  ```

where a.long1 is a LONG column.

Functions

The following restrictions apply to using functions:

- Unsupported functions cannot be used in statements that refer to LONG columns.
- When negative numbers are used as the second parameter in a SUBSTR function, incorrect results are returned. This is due to incompatibility between the Oracle SUBSTR function and the equivalent in SQL Server.

SQL*Plus COPY Command with Lowercase Table Names

You need to use double quotes to wrap around lowercase table names.

For example:

```
copy from tkhouser/tkhouser@inst1 insert loc_tkhodept using select * from
"tkhodept"@holink2;
```

Database Links

The gateway is not multithreaded and cannot support shared database links. Each gateway session spawns a separate gateway process and connections cannot be shared.

Known Problems

This section describes known problems and includes suggestions for correcting them when possible. If you have any questions or concerns about the problems, contact Oracle Support Services. A current list of problems is available online. Contact your local Oracle office for information about accessing the list.

The following known problems are described in this section:

- Encrypted Format Login

Encrypted Format Login

The Oracle9*i* database (Release 9.2 and earlier) supported an Oracle initialization parameter, DBLINK_ENCRYPT_LOGIN. When this parameter is set to TRUE, the password for the login user ID is not sent over the network.

If this parameter is set to TRUE in the initialization parameter file used by the Oracle9*i* database, you must change the setting to FALSE, the default setting, to allow Oracle9*i* to communicate with the gateway.

In the current release, Oracle Database 11*g*, Release 11.1, the DBLINK_ENCRYPT_LOGIN initialization parameter is obsolete, so you need not check it.

Date Arithmetic

The following SQL expressions do not function correctly with the gateway:

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

Statements with the preceding expressions are sent to the SQL Server database without any translation. Since SQL Server does not support these date arithmetic functions, the statements return an error.

SQL Server IMAGE, TEXT and NTEXT Data Types

The following restrictions apply when using IMAGE, TEXT, and NTEXT data types:

- An unsupported SQL function cannot be used in a SQL statement that accesses a column defined as SQL Server data type IMAGE, TEXT, or NTEXT.
- You cannot use SQL*Plus to select data from a column defined as SQL Server data type IMAGE, TEXT, or NTEXT when the data is greater than 80 characters in length. Oracle recommends using Pro*C or Oracle Call Interface to access such data in a SQL Server database.
- IMAGE, TEXT, and NTEXT data cannot be read through pass-through queries.
- If a SQL statement is accessing a table including an IMAGE, TEXT, or NTEXT column, the statement will be sent to SQL Server as two separate statements. One statement to access the IMAGE, TEXT or NTEXT column, and a second statement for the other columns in the original statement. This will result in two connections to SQL Server due to a limitation in the Microsoft ODBC driver which only allows one statement for each connection, which can cause a hang depending on the sequence of SQL statements. If this happens, try issuing a commit and separating the statements in different transactions.

The gateway does not support the PL/SQL function `COLUMN_VALUE_LONG` of the `DBMS_SQL` package.

> **See Also:** Appendix B, "Supported SQL Syntax and Functions" for more information about restrictions on SQL syntax.

String Functions

If you concatenate numeric literals using the "||" or `CONCAT` operator when using the gateway to query a SQL Server database, the result is an arithmetic addition. For example, the result of the following statement is 18:

```
SQL> SELECT 9 || 9 FROM DUAL@MSQL;
```

The result is 99 when using Oracle to query an Oracle database.

Schema Names and PL/SQL

If you do not prefix a SQL Server database object with its schema name in a SQL statement within a PL/SQL block, the following error message occurs:

```
ORA-6550 PLS-201 Identifier table_name must be declared.
```

Change the SQL statement to include the schema name of the object.

Data Dictionary Views and PL/SQL

You cannot refer to data dictionary views in SQL statements that are inside a PL/SQL block.

Stored Procedures

Return values of stored procedures which return result sets are incorrect.

3

Case Studies

The following case studies for SQL Server demonstrate some of the features of the Oracle Database Gateway. You can verify that the gateway is installed and operating correctly by using the demonstration files included in the distribution media.

The demonstration files are automatically copied to disk when the gateway is installed.

This chapter contains the following sections:

Case Descriptions

The cases illustrate:

- A simple query (Case 1)
- A more complex query (Case 2)
- Joining SQL Server tables (Case 3)
- Write capabilities (Case 4)
- A data dictionary query (Case 5)
- The pass-through feature (Case 6)
- Executing stored procedures (Case 7)

Installation Media Contents

The installation media contains the following:

- Demonstration files
- One SQL script file that creates the demonstration tables and stored procedures in the SQL Server database
- One SQL script file that drops the demonstration tables and stored procedures from the SQL Server database

Demonstration Files

After a successful gateway installation, use the demonstration files stored in the directory `ORACLE_HOME\dg4msql\demo` where `ORACLE_HOME` is the directory under which the gateway is installed. The directory contains the following demonstration files:

Demonstration Files
bldmsql.sql
case1.sql
case2.sql
case3.sql
case4a.sql
case4b.sql
case4c.sql
case5.sql
case6a.sql
case6b.sql
case7.sql
dropmsql.sql

Demonstration Requirements

The case studies assume these requirements have been met:

- The gateway demonstration tables and stored procedures are installed in the SQL Server database
- The Oracle database has an account named `SCOTT` with a password of `TIGER`
- The Oracle database has a database link called `GTWLINK` (set up as public or private to the user `SCOTT`) which connects the gateway to a SQL Server database as `SCOTT` with password `TIGER2`

  For example, you can create the database link as follows:

```
SQL> CREATE DATABASE LINK GTWLINK CONNECT TO SCOTT
  2    IDENTIFIED BY TIGER2 USING 'GTWSID';
```

- Oracle Net Services is configured correctly and running

Creating Demonstration Tables

The case studies are based on the GTW_EMP, GTW_DEPT, and GTW_SALGRADE tables and the stored procedures InsertDept and GetDept. If the demonstration tables and stored procedures have not been created in the SQL Server database, use the bldmsql.sql script to create them. Enter the following:

```
> isql -USCOTT -PTIGER2 -ibldmsql.sql
```

The script creates the demonstration tables and stored procedures in the SQL Server database accordingly:

```
CREATE TABLE GTW_EMP (
EMPNO      SMALLINT NOT NULL
ENAME      VARCHAR(10),
JOB        VARCHAR(9),
MGR        SMALLINT,
HIREDATE   DATETIME,
SAL        NUMERIC(7,2),
COMM       NUMERIC(7,2),
DEPTNO     SMALLINT)
go

CREATE TABLE GTW_DEPT (
DEPTNO     SMALLINT NOT NULL,
DNAME      VARCHAR(14),
LOC        VARCHAR(13))
go

CREATE TABLE GTW_SALGRADE (
GRADE      MONEY,
LOSAL      NUMERIC(9,4),
HISAL      NUMERIC(9,4))
go

DROP PROCEDURE InsertDept
go

CREATE PROCEDURE InsertDept (@dno INTEGER,
   @dname VARCHAR(14), @loc VARCHAR(13))
AS INSERT INTO GTW_DEPT VALUES (@dno, @dname, @loc)
go

DROP PROCEDURE GetDept
go

CREATE PROCEDURE GetDept (@dno INTEGER, @dname VARCHAR(14) OUTPUT)
AS SELECT @dname=DNAME FROM GTW_DEPT WHERE DEPTNO=@dno
go
```

Demonstration Table Definitions

The following table definitions use information retrieved by the SQL*PLUS DESCRIBE command:

GTW_EMP

```
Name                          Null?    Type
----------------------------- -------- ----
EMPNO                         NOT NULL NUMBER(5)
ENAME                                  VARCHAR2(10)
```

```
JOB                                       VARCHAR2(9)
MGR                                       NUMBER(5)
HIREDATE                                  DATE
SAL                                       NUMBER(7,2)
COMM                                      NUMBER(7,2)
DEPTNO                                    NUMBER(5)
```

GTW_DEPT

```
Name                           Null?    Type
------------------------------ -------- ----
DEPTNO                         NOT NULL NUMBER(5)
DNAME                                   VARCHAR2(14)
LOC                                     VARCHAR2(13)
```

GTW_SALGRADE

```
Name                           Null?    Type
------------------------------ -------- ----
GRADE                                   NUMBER(19,4)
LOSAL                                   NUMBER(9,4)
HISAL                                   NUMBER(9,4)
```

Demonstration Table Contents

The contents of the SQL Server tables are:

GTW_EMP

```
EMPNO ENAME   JOB         MGR   HIREDATE   SAL   COMM   DEPTNO
----- -----   ---         ---   --------   ---   ----   ------
7369  SMITH   CLERK       7902  17-DEC-80   800             20
7499  ALLEN   SALESMAN    7698  20-FEB-81  1600   300       30
7521  WARD    SALESMAN    7698  22-FEB-81  1250   500       30
7566  JONES   MANAGER     7839  02-APR-81  2975             20
7654  MARTIN  SALESMAN    7698  28-SEP-81  1250  1400       30
7698  BLAKE   MANAGER     7839  01-MAY-81  2850             30
7782  CLARK   MANAGER     7839  09-JUN-81  2450             10
7788  SCOTT   ANALYST     7566  09-DEC-82  3000             20
7839  KING    PRESIDENT         17-NOV-81  5000             10
7844  TURNER  SALESMAN    7698  08-SEP-81  1500     0       30
7876  ADAMS   CLERK       7788  12-JAN-83  1100             20
7900  JAMES   CLERK       7698  03-DEC-81   950             30
7902  FORD    ANALYST     7566  03-DEC-81  3000             20
7934  MILLER  CLERK       7782  23-JAN-82  1300             10
```

GTW_DEPT

```
DEPTNO DNAME          LOC
------ -------------- --------
   10 ACCOUNTING     NEW YORK
   20 RESEARCH       DALLAS
   30 SALES          CHICAGO
   40 OPERATIONS     BOSTON
```

GTW_SALGRADE

```
GRADE      LOSAL      HISAL
------     ------     -----
    1        700      1200
    2       1201      1400
    3       1401      2000
```

```
4       2001      3000
5       3001      9999
```

Case 1: Simple Queries

Case 1 demonstrates the following:

- A simple query
- A simple query retrieving full date information

The first query retrieves all the data from `GTW_DEPT` and confirms that the gateway is working correctly. The second query retrieves all the data from `GTW_EMP` including the time portion of the hire date because the default date format was set to `DD-MON-YY HH24:MM:SS` for the session by an `ALTER SESSION` command.

Case 2: A More Complex Query

Case 2 demonstrates the following:

- The functions `SUM(`*`expression`*`)` and `NVL(`*`expr1, expr2`*`)` in the `SELECT` list
- The `GROUP BY` and `HAVING` clauses

This query retrieves the departments from `GTW_EMP` whose total monthly expenses are higher than `$10,000`.

Case 3: Joining SQL Server Tables

Case 3 demonstrates the following:

- Joins between SQL Server tables
- Subselects

The query retrieves information from three SQL Server tables and relates the employees to their department name and salary grade, but only for those employees earning more than the average salary.

Case 4: Write Capabilities

Case 4 is split into three cases and demonstrates the following:

- DELETE Statement
- UPDATE Statement
- INSERT Statement

DELETE Statement

Case 4a demonstrates bind values and subselect. All employees in department `20` and one employee, `WARD`, in department `30` are deleted.

UPDATE Statement

Case 4b provides an example of a simple `UPDATE` statement. In this example, employees are given a `$100` a month salary increase.

INSERT Statement

Case 4c is an example of a simple insert statement that does not provide information for all columns.

Case 5: Data Dictionary Query

Case 5 demonstrates data dictionary mapping. It retrieves all the tables and views that exist in the SQL Server database that begin with `GTW`.

Case 6: The Pass-Through Feature

Case 6 demonstrates the gateway pass-through feature which allows an application to send commands or statements to SQL Server.

This case demonstrates:

- A pass-through `UPDATE` statement using bind variables
- A pass-through `SELECT` statement

UPDATE Statement

Case 6a provides an example of a pass-through `UPDATE` statement with bind variables. In this example, the salary for `EMPNO 7934` is set to `4000`.

SELECT Statement

Case 6b provides an example of a pass-through `SELECT` statement. The data that is returned from the `SELECT` statement is inserted into a local table at the Oracle database.

Case 7: Executing Stored Procedures

Case 7 demonstrates the gateway executing a stored procedure in the SQL Server database.

A

Data Type Conversion

This appendix contains the following section:

- Data Type Conversion

Data Type Conversion

The gateway converts SQL Server data types to Oracle data types as follows:

Table A–1 Data Type Conversions

SQL Server	Oracle	Comment
BIGINT	NUMBER(19)	
BIGINT IDENTITY	NUMBER(19)	
BINARY	RAW	-
BIT	NUMBER(3)	-
CHAR	CHAR	-
DATETIME	DATE	Fractional parts of a second are truncated
DECIMAL	NUMBER(p[,s])	-
DECIMAL IDENTITY	NUMBER(p[,s])	
FLOAT	FLOAT(53)	-
IMAGE	LONG RAW	-
INT	NUMBER(10)	
INT IDENTITY	NUMBER(10)	
MONEY	NUMBER(19,4)	-
NCHAR	NCHAR	-
NTEXT	LONG	if Oracle DB Character Set = Unicode. Otherwise, it is not supported
NVARCHAR	NVARCHAR	-
NVARCHAR(MAX)	LONG	if Oracle DB Character Set = Unicode. Otherwise, it is not supported
NUMERIC	NUMBER(p[,s])	-
NUMERIC IDENTITY	NUMBER(p[,s])	
REAL	FLOAT(24)	-
SMALLDATETIME	DATE	-

Table A–1 (Cont.) Data Type Conversions

SQL Server	Oracle	Comment
SMALLMONEY	NUMBER(10,4)	-
SMALLINT	NUMBER(5)	-
SMALLINT IDENTITY	NUMBER(5)	
SYSNAME	NVARCHAR	-
TEXT	LONG	
TIMESTAMP	RAW	-
TINYINT	NUMBER(3)	-
TINYINT IDENTITY	NUMBER(3)	
VARBINARY	RAW	-
VARBINARY(MAX)	LONG RAW	
VARCHAR	VARCHAR2	-
VARCHAR(MAX)	LONG	-
XML	LONG	

B

Supported SQL Syntax and Functions

This appendix contains the following sections:

Supported SQL Statements

With a few exceptions, the gateway provides full support for Oracle `DELETE`, `INSERT`, `SELECT`, and `UPDATE` statements.

The gateway does not support Oracle data definition language (DDL) statements. No form of the Oracle `ALTER`, `CREATE`, `DROP`, `GRANT`, or `TRUNCATE` statements can be used. Instead, use the pass-through feature of the gateway if you need to use DDL statements against the SQL Server database.

Note: `TRUNCATE` cannot be used in a pass-through statement.

See Also: *Oracle Database Reference* for a detailed descriptions of keywords, parameters, and options.

DELETE

The `DELETE` statement is fully supported. However, only Oracle functions supported by SQL Server can be used.

See Also: "Functions Supported by SQL Server" on page B-2 for a list of supported functions.

INSERT

The `INSERT` statement is fully supported. However, only Oracle functions supported by SQL Server can be used.

See Also: "Functions Supported by SQL Server" on page B-2 for a list of supported functions.

SELECT

The `SELECT` statement is fully supported, with these exceptions:

- CONNECT BY *condition*
- NOWAIT
- START WITH *condition*
- WHERE CURRENT OF

UPDATE

The UPDATE statement is fully supported. However, only Oracle functions supported by SQL Server can be used.

> **See Also:** "Functions Supported by SQL Server" on page B-2 for a list of supported functions.

Oracle Functions

All functions are evaluated by the SQL Server database after the gateway has converted them to SQL Server SQL equivalents. The exception is the TO_DATE function, which is evaluated by the gateway.

Functions Not Supported by SQL Server

Oracle SQL functions with no equivalent function in SQL Server are not supported in DELETE, INSERT, or UPDATE statements, but are evaluated by the Oracle database if the statement is a SELECT statement. That is, the Oracle database performs post-processing of SELECT statements sent to the gateway.

If an unsupported function is used in a DELETE, INSERT, or UPDATE, statement, the following Oracle error occurs:

```
ORA-02070: database db_link_name does not support function in this context
```

Functions Supported by SQL Server

The gateway translates the following Oracle database functions in SQL statements to their equivalent SQL Server functions:

- Arithmetic Operators
- Comparison Operators
- Pattern Matching
- Group Functions
- String Functions
- Other Functions

Arithmetic Operators

Oracle	SQL Server
+	+
-	-
*	*
/	/

Comparison Operators

Oracle	SQL Server
=	=
>	>
<	<
>=	>=
<=	<=
<>, !=, ^=	<>
IS NOT NULL	IS NOT NULL
IS NULL	IS NULL

Pattern Matching

Oracle	SQL Server
LIKE	LIKE
NOT LIKE	NOT LIKE

Group Functions

Oracle	SQL Server
AVG	AVG
COUNT	COUNT
MAX	MAX
MIN	MIN
SUM	SUM

String Functions

Oracle	SQL Server
||, CONCAT	+ (*expression1* + *expression2*)
ASCII	ASCII
CHR	CHAR
INSTR (with two arguments)	CHARINDEX
LENGTH	DATALENGTH
LOWER	LOWER
LTRIM	LTRIM
RTRIM	RTRIM
SUBSTR (second argument cannot be a negative number)	SUBSTRING
UPPER	UPPER

Other Functions

Oracle	SQL Server
ABS	ABS
CEIL	CEILING
COS	COS
EXP	EXP
FLOOR	FLOOR
LN	LOG
LOG	LOG10
MOD	%
NOT NVL	IS NOT NULL
NVL	IS NULL
POWER	POWER
ROUND	ROUND
SIN	SIN
SQRT	SQRT
TAN	TAN

Functions Supported by the Gateway

If an Oracle function has no equivalent function in SQL Server, the Oracle function is not translated into the SQL statement and must be post-processed if the SQL statement is a SELECT.

The gateway, however, does support the TO_DATE function equivalent in SQL Server, as follows:

TO_DATE(*date_string* | *date_column*)
Where:

date_string is converted to a string with the following format:

yyyy-mm-dd hh:*mi*:*ss*.*fff*

Recommendation: Supply the date string with the same format as the result (that is, *yyyyy-mm-dd hh*:*mi*:*ss*.*fff*).

---

date_column is a column with a date data type. It is converted to a parameter with a timestamp data type.

C

Data Dictionary

The Oracle Database Gateway for SQL Server translates a query that refers to an Oracle database data dictionary table into a query that retrieves the data from SQL Server system tables. You perform queries on data dictionary tables over the database link in the same way you query data dictionary tables in the Oracle database. The gateway data dictionary is similar to the Oracle database data dictionary in appearance and use.

This appendix contains the following sections:

Data Dictionary Support

The following paragraphs describe the Oracle Database Gateway for SQL Server data dictionary support.

SQL Server System Tables

SQL Server data dictionary information is stored in the SQL Server database as SQL Server system tables. All SQL Server system tables have names prefixed with "`sys`". The SQL Server system tables define the structure of a database. When you change data definitions, SQL Server reads and modifies the SQL Server system tables to add information about the user tables.

Accessing the Gateway Data Dictionary

Accessing a gateway data dictionary table or view is identical to accessing a data dictionary in an Oracle database. You issue a SQL `SELECT` statement specifying a database link. The Oracle database data dictionary view and column names are used to access the gateway data dictionary in an Oracle database. Synonyms of supported views are also acceptable. For example, the following statement queries the data dictionary table `ALL_CATALOG` to retrieve all table names in the SQL Server database:

```
SQL> SELECT * FROM "ALL_CATALOG"@MSQL;
```

When a data dictionary access query is issued, the gateway:

1. Maps the requested table, view, or synonym to one or more SQL Server system table names. The gateway translates all data dictionary column names to their corresponding SQL Server column names within the query. If the mapping involves one SQL Server system table, the gateway translates the requested table

name to its corresponding SQL Server system table name within the query. If the mapping involves multiple SQL Server system tables, the gateway constructs a join in the query using the translated SQL Server system table names.

2. Sends the translated query to SQL Server.
3. Might convert the retrieved SQL Server data to give it the appearance of the Oracle database data dictionary table.
4. Passes the data dictionary information from the translated SQL Server system table to the Oracle database.

Note: The values returned when querying the gateway data dictionary might not be the same as the ones returned by the Oracle SQL*Plus `DESCRIBE` command.

Direct Queries to SQL Server Tables

Queries issued directly to individual SQL Server system tables are allowed but they return different results because the SQL Server system table column names differ from those of the data dictionary view. Also, certain columns in an SQL Server system table cannot be used in data dictionary processing.

Supported Views and Tables

The gateway supports the following views and tables:

Supported Views and Table	Supported Views and Table
ALL_CATALOG	ALL_COL_COMMENTS
ALL_CONS_COLUMNS	ALL_CONSTRAINTS
ALL_IND_COLUMNS	ALL_INDEXES
ALL_OBJECTS	ALL_TAB_COLUMNS
ALL_TAB_COMMENTS	ALL_TABLES
ALL_USERS	ALL_VIEWS
DBA_CATALOG	DBA_COL_COMMENTS
DBA_OBJECTS	DBA_TAB_COLUMNS
DBA_TAB_COMMENTS	DBA_TABLES
DICT_COLUMNS	DICTIONARY
DUAL	TABLE_PRIVILEGES
USER_CATALOG	USER_COL_COMMENTS
USER_CONS_COLUMNS	USER_CONSTRAINTS
USER_IND_COLUMNS	USER_INDEXES
USER_OBJECTS	USER_TAB_COLUMNS
USER_TAB_COMMENTS	USER_TABLES
USER_USERS	USER_VIEWS

No other Oracle database data dictionary tables or views are supported. If you use a view not on the list, you will receive the Oracle database error code for no more rows available.

Queries through the gateway of any data dictionary table or view beginning with ALL_ can return rows from the SQL Server database even when access privileges for those SQL Server objects have not been granted. When querying an Oracle database with the Oracle data dictionary, rows are returned only for those objects you are permitted to access.

Data Dictionary Mapping

The tables in this section list Oracle data dictionary view names and the equivalent SQL Server system tables used. A plus sign (+) indicates that a join operation is involved.

Table C–1 Oracle Data Dictionary View Names and SQL Server Equivalents

View Name	SQL Server System Table Name
ALL_CATALOG	sysusers + sysobjects
ALL_COL_COMMENTS	sysusers + sysobjects + syscolumns
ALL_CONS_COLUMNS	sp_pkeys + sp_fkeys
ALL_CONSTRAINTS	sysusers + sysobjects + sysindexes + sysconstraints + sysreferences
ALL_IND_COLUMNS	sysusers + sysindexes + syscolumns
ALL_INDEXES	sysusers + sysindexes + sysobjects
ALL_OBJECTS	sysusers + sysobjects + sysindexes
ALL_TAB_COLUMNS	sysusers + sysobjects + syscolumns
ALL_TAB_COMMENTS	sysusers + sysobjects
ALL_TABLES	sysusers + sysobjects
ALL_USERS	sysusers
ALL_VIEWS	sysusers + sysobjects + syscomments
DBA_CATALOG	sysusers + sysobjects
DBA_COL_COMMENTS	sysusers + sysobjects + syscolumns
DBA_OBJECTS	sysusers + sysobjects + sysindexes
DBA_TABLES	sysusers + sysobjects
DBA_TAB_COLUMNS	sysusers + sysobjects + syscolumns
DBA_TAB_COMMENTS	sysusers + sysobjects
DICT_COLUMNS	sysobjects + syscolumns
DICTIONARY	sysobjects
DUAL	sysusers
TABLE_PRIVILEGES	sysprotects + sysusers + sysobjects
USER_CATALOG	sysusers + sysobjects
USER_COL_COMMENTS	sysusers + sysobjects + syscolumns
USER_CONS_COLUMNS	sp_pkeys + sp_fkeys

Table C–1 (Cont.) Oracle Data Dictionary View Names and SQL Server Equivalents

View Name	SQL Server System Table Name
USER_CONSTRAINTS	sysusers + sysobjects + sysindexes + sysconstraints + sysreferences
USER_IND_COLUMNS	sysusers + sysindexes + syscolumns
USER_INDEXES	sysusers + sysindexes + sysobjects
USER_OBJECTS	sysusers + sysobjects + sysindexes
USER_TAB_COLUMNS	sysusers + sysobjects + syscolumns
USER_TAB_COMMENTS	sysusers + sysobjects
USER_TABLES	sysusers + sysobjects
USER_USERS	sysusers
USER_VIEWS	sysusers + sysobjects + syscomments

Default Column Values

There is a minor difference between the gateway data dictionary and a typical Oracle database data dictionary. The Oracle database columns that are missing in an SQL Server system table are filled with zeros, spaces, null values, not-applicable values (N.A.), or default values, depending on the column type.

Gateway Data Dictionary Descriptions

The gateway data dictionary tables and views provide the following information:

- Name, data type, and width of each column
- The contents of columns with fixed values

They are described here with information retrieved by an Oracle SQL*Plus `DESCRIBE` command. The values in the `Null?` column might differ from the Oracle database data dictionary tables and views. Any default value is shown to the right of an item, but this is not information returned by `DESCRIBE`.

Note: The column width of some columns in the translated data dictionary tables would be different when the gateway connects to a SQL Server Version 7.0 database.

Table C–2 ALL_CATALOG

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
TABLE_TYPE	-	VARCHAR2(5)	"TABLE" or "VIEW"

Table C–3 ALL_COL_COMMENTS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
COLUMN_NAME	-	VARCHAR2(256)	-
COMMENTS	-	VARCHAR2(1)	-

Table C–4 ALL_CONS_COLUMNS

Name	Null?	Type	Value
OWNER	NOT NULL	VARCHAR2(30)	-
CONSTRAINT_NAME	-	VARCHAR2(30)	-
TABLE_NAME	-	VARCHAR2(30)	-
COLUMN_NAME	-	VARCHAR2(8192)	-
POSITION	-	FLOAT(53)	-

Table C–5 ALL_CONSTRAINTS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
CONSTRAINT_NAME	-	VARCHAR2(256)	-
CONSTRAINT_TYPE	-	VARCHAR2(1)	"C" or "P" or "R" or "U"
TABLE_NAME	-	VARCHAR2(256)	-
SEARCH_CONDITION	-	VARCHAR2(1)	NULL
R_OWNER	-	VARCHAR2(256)	-
R_CONSTRAINT_NAME	-	VARCHAR2(256)	-
DELETE_RULE	-	VARCHAR2(1)	NULL
STATUS	-	VARCHAR2(1)	NULL
DEFERRABLE	-	VARCHAR2(1)	NULL
DEFERRED	-	VARCHAR2(1)	NULL
VALIDATED	-	VARCHAR2(1)	NULL
GENERATED	-	VARCHAR2(1)	NULL
BAD	-	VARCHAR2(1)	NULL
RELY	-	VARCHAR2(1)	NULL
LAST_CHANGE	-	DATE	-

Table C–6 ALL_IND_COLUMNS

Name	Null?	Type	Value
INDEX_OWNER	NOT NULL	VARCHAR2(30)	-
INDEX_NAME	NOT NULL	VARCHAR2(30)	-
TABLE_OWNER	NOT NULL	VARCHAR2(30)	-
TABLE_NAME	NOT NULL	VARCHAR2(30)	-
COLUMN_NAME	-	VARCHAR2(8192)	-
COLUMN_POSITION	NOT NULL	FLOAT(53)	-
COLUMN_LENGTH	NOT NULL	FLOAT(53)	-
DESCEND	-	VARCHAR2(4)	-

Table C–7 ALL_INDEXES

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
INDEX_NAME	-	VARCHAR2(256)	-
INDEX_TYPE	-	VARCHAR2(1)	NULL
TABLE_OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
TABLE_TYPE	-	VARCHAR(7)	"TABLE" or "CLUSTER"
UNIQUENESS	-	VARCHAR2(1)	NULL
COMPRESSION	-	VARCHAR2(1)	NULL
PREFIX_LENGTH	-	NUMBER	0
TABLESPACE_NAME	-	VARCHAR2(1)	NULL
INI_TRANS	-	NUMBER	0
MAX_TRANS	-	NUMBER	0
INITIAL_EXTENT	-	NUMBER	0
NEXT_EXTENT	-	NUMBER	0
MIN_EXTENTS	-	NUMBER	0
MAX_EXTENTS	-	NUMBER	0
PCT_INCREASE	-	NUMBER	0
PCT_THRESHOLD	-	NUMBER	0
INCLUDE_COLUMN	-	NUMBER	0
FREELISTS	-	NUMBER	0
FREELIST_GROUPS	-	NUMBER	0
PCT_FREE	-	NUMBER	0
LOGGING	-	VARCHAR2(1)	NULL
BLEVEL	-	NUMBER	0

Table C–7 (Cont.) ALL_INDEXES

Name	Null?	Type	Value
LEAF_BLOCKS	-	NUMBER	0
DISTINCT_KEYS	-	NUMBER	0
AVG_LEAF_BLOCKS_PER_KEY	-	NUMBER	0
AVG_DATA_BLOCKS_PER_KEY	-	NUMBER	0
CLUSTERING_FACTOR	-	NUMBER	0
STATUS	-	VARCHAR2(1)	NULL
NUM_ROWS	-	NUMBER	0
SAMPLE_SIZE	-	NUMBER	0
LAST_ANALYZED	-	DATE	NULL
DEGREE	-	VARCHAR2(1)	NULL
INSTANCES	-	VARCHAR2(1)	NULL
PARTITIONED	-	VARCHAR2(1)	NULL
TEMPORARY	-	VARCHAR2(1)	NULL
GENERATED	-	VARCHAR2(1)	NULL
SECONDARY	-	VARCHAR2(1)	NULL
BUFFER_POOL	-	VARCHAR2(1)	NULL
USER_STATS	-	VARCHAR2(1)	NULL
DURATION	-	VARCHAR2(1)	NULL
PCT_DIRECT_ACCESS	-	NUMBER	0
ITYP_OWNER	-	VARCHAR2(1)	NULL
ITYP_NAME	-	VARCHAR2(1)	NULL
PARAMETERS	-	VARCHAR2(1)	NULL
GLOBAL_STATS	-	VARCHAR2(1)	NULL
DOMIDX_STATUS	-	VARCHAR2(1)	NULL
DOMIDX_OPSTATUS	-	VARCHAR2(1)	NULL
FUNCIDX_STATUS	-	VARCHAR2(1)	NULL

Table C–8 ALL_OBJECTS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
OBJECT_NAME	-	VARCHAR2(256)	-
SUBOBJECT_NAME	-	VARCHAR2(1)	NULL
OBJECT_ID	-	NUMBER	-
DATA_OBJECT_ID	-	NUMBER	0

Table C–8 (Cont.) ALL_OBJECTS

Name	Null?	Type	Value
OBJECT_TYPE	-	VARCHAR2(9)	"TABLE" or "VIEW" or "INDEX" or "PROCEDURE"
CREATED	-	DATE	-
LAST_DDL_TIME	-	DATE	-
TIMESTAMP	-	VARCHAR2(1)	NULL
STATUS	-	VARCHAR2(5)	"VALID"
TEMPORARY	-	VARCHAR2(1)	NULL
GENERATED	-	VARCHAR2(1)	NULL
SECONDARY	-	VARCHAR2(1)	NULL

Table C–9 ALL_TAB_COLUMNS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
COLUMN_NAME	-	VARCHAR2(256)	-
DATA_TYPE	-	VARCHAR2(8)	-
DATA_TYPE_MOD	-	VARCHAR2(1)	NULL
DATA_TYPE_OWNER	-	VARCHAR2(1)	NULL
DATA_LENGTH	-	NUMBER	-
DATA_PRECISION	-	NUMBER	-
DATA_SCALE	-	NUMBER	-
NULLABLE	-	VARCHAR2(1)	"Y" or "N"
COLUMN_ID	-	NUMBER	-
DEFAULT_LENGTH	-	NUMBER	0
DATA_DEFAULT	-	VARCHAR2(1)	NULL
NUM_DISTINCT	-	NUMBER	0
LOW_VALUE	-	NUMBER	0
HIGH_VALUE	-	NUMBER	0
DENSITY	-	NUMBER	0
NUM_NULLS	-	NUMBER	0
NUM_BUCKETS	-	NUMBER	0
LAST_ANALYZED	-	DATE	NULL
SAMPLE_SIZE	-	NUMBER	0
CHARACTER_SET_NAME	-	VARCHAR2(1)	NULL
CHAR_COL_DEC_LENGTH	-	NUMBER	0

Table C–9 (Cont.) ALL_TAB_COLUMNS

Name	Null?	Type	Value
GLOBAL_STATS	-	VARCHAR2(1)	NULL
USER_STATS	-	VARCHAR2(1)	NULL
AVG_COL_LEN	-	NUMBER	0

Table C–10 ALL_TAB_COMMENTS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
TABLE_TYPE	-	VARCHAR2(5)	"TABLE" or "VIEW"
COMMENTS	-	VARCHAR2(1)	NULL

Table C–11 ALL_TABLES

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
TABLESPACE_NAME	-	VARCHAR2(1)	NULL
CLUSTER_NAME	-	VARCHAR2(1)	NULL
IOT_NAME	-	VARCHAR2(1)	NULL
PCT_FREE	-	NUMBER	0
PCT_USED	-	NUMBER	0
INI_TRANS	-	NUMBER	0
MAX_TRANS	-	NUMBER	0
INITIAL_EXTENT	-	NUMBER	0
NEXT_EXTENT	-	NUMBER	0
MIN_EXTENTS	-	NUMBER	0
MAX_EXTENTS	-	NUMBER	0
PCT_INCREASE	-	NUMBER	0
FREELISTS	-	NUMBER	0
FREELIST_GROUPS	-	NUMBER	0
LOGGING	-	VARCHAR2(1)	NULL
BACKED_UP	-	VARCHAR2(1)	NULL
NUM_ROWS	-	NUMBER	0
BLOCKS	-	NUMBER	0
EMPTY_BLOCKS	-	NUMBER	0

Table C–11 (Cont.) ALL_TABLES

Name	Null?	Type	Value
AVG_SPACE	-	NUMBER	0
CHAIN_CNT	-	NUMBER	0
AVG_ROW_LEN	-	NUMBER	0
AVG_SPACE_FREELIST_BLOCKS	-	NUMBER	0
NUM_FREELIST_BLOCKS	-	NUMBER	0
DEGREE	-	VARCHAR2(1)	NULL
INSTANCES	-	VARCHAR2(1)	NULL
CACHE	-	VARCHAR2(1)	NULL
TABLE_LOCK	-	VARCHAR2(1)	NULL
SAMPLE_SIZE	-	NUMBER	0
LAST_ANALYZED	-	DATE	NULL
PARTITIONED	-	VARCHAR2(1)	NULL
IOT_TYPE	-	VARCHAR2(1)	NULL
TEMPORARY	-	VARHCAR2(1)	NULL
SECONDARY	-	VARCHAR2(1)	NULL
NESTED	-	VARCHAR2(1)	NULL
BUFFER_POOL	-	VARCHAR2(1)	NULL
ROW_MOVEMENT	-	VARCHAR2(1)	NULL
GLOBAL_STATS	-	VARCHAR2(1)	NULL
USER_STATS	-	VARCHAR2(1)	NULL
DURATION	-	VARHCAR2(1)	NULL
SKIP_CORRUPT	-	VARCHAR2(1)	NULL
MONITORING	-	VARCHAR2(1)	NULL

Table C–12 ALL_USERS

Name	Null?	Type	Value
USERNAME	-	VARCHAR2(256)	-
USER_ID	NOT NULL	NUMBER(5)	-
CREATED	NOT NULL	DATE	-

Table C–13 ALL_VIEWS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
VIEW_NAME	-	VARCHAR2(256)	-
TEXT_LENGTH	-	NUMBER	0

Table C–13 (Cont.) ALL_VIEWS

Name	Null?	Type	Value
TEXT	-	VARCHAR2(256)	-
TYPE_TEXT_LENGTH	-	NUMBER	0
TYPE_TEXT	-	VARCHAR2(1)	-
OID_TEXT_LENGTH	-	NUMBER	0
OID_TEXT	-	VARCHAR2(1)	-
VIEW_TYPE_OWNER	-	VARCHAR2(1)	-
VIEW_TYPE	-	VARCHAR2(1)	-

Table C–14 DBA_CATALOG

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
TABLE_TYPE	-	VARCHAR2(5)	"TABLE" or "VIEW"

Table C–15 DBA_COL_COMMENTS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
COLUMN_NAME	-	VARCHAR2(256)	-
COMMENTS	-	VARCHAR2(1)	NULL

Table C–16 DBA_OBJECTS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
OBJECT_NAME	-	VARCHAR2(256)	-
SUBOBJECT_NAME	-	VARCHAR2(1)	NULL
OBJECT_ID	-	NUMBER	-
DATA_OBJECT_ID	-	NUMBER	0
OBJECT_TYPE	-	VARCHAR2(9)	"TABLE" or "VIEW" or "INDEX" or "PROCEDURE"
CREATED	-	DATE	-
LAST_DDL_TIME	-	DATE	-

Table C–16 (Cont.) DBA_OBJECTS

Name	Null?	Type	Value
TIMESTAMP	-	VARCHAR2(1)	NULL
STATUS	-	VARCHAR2(5)	NULL
TEMPORARY	-	VARCHAR2(1)	NULL
GENERATED	-	VARCHAR2(1)	NULL
SECONDARY	-	VARCHAR2(1)	NULL

Table C–17 DBA_TAB_COLUMNS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
COLUMN_NAME	-	VARCHAR2(256)	-
DATA_TYPE	-	VARCHAR2(8)	-
DATA_TYPE_MOD	-	VARCHAR2(1)	NULL
DATA_TYPE_OWNER	-	VARCHAR2(1)	NULL
DATA_LENGTH	-	NUMBER	-
DATA_PRECISION	-	NUMBER	-
DATA_SCALE	-	NUMBER	-
NULLABLE	-	VARCHAR2(1)	"Y" or "N"
COLUMN_ID	-	NUMBER	-
DEFAULT_LENGTH	-	NUMBER	0
DATA_DEFAULT	-	VARCHAR2(1)	NULL
NUM_DISTINCT	-	NUMBER	0
LOW_VALUE	-	NUMBER	0
HIGH_VALUE	-	NUMBER	0
DENSITY	-	NUMBER	0
NUM_NULLS	-	NUMBER	0
NUM_BUCKETS	-	NUMBER	0
LAST_ANALYZED	-	DATE	NULL
SAMPLE_SIZE	-	NUMBER	0
CHARACTER_SET_NAME	-	VARCHAR2(1)	NULL
CHAR_COL_DEC_LENGTH	-	NUMBER	0
GLOBAL_STATS	-	VARCHAR2(1)	NULL
USER_STATS	-	VARCHAR2(1)	NULL
AVG_COL_LEN	-	NUMBER	0

Table C–18 DBA_TAB_COMMENTS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
TABLE_TYPE	-	VARCHAR2(5)	"TABLE" or "VIEW"
COMMENTS	-	VARCHAR2(1)	NULL

Table C–19 DBA_TABLES

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
TABLESPACE_NAME	-	VARCHAR2(1)	NULL
CLUSTER_NAME	-	VARCHAR2(1)	NULL
IOT_NAME	-	VARCHAR2(1)	NULL
PCT_FREE	-	NUMBER	0
PCT_USED	-	NUMBER	0
INI_TRANS	-	NUMBER	0
MAX_TRANS	-	NUMBER	0
INITIAL_EXTENT	-	NUMBER	0
NEXT_EXTENT	-	NUMBER	0
MIN_EXTENTS	-	NUMBER	0
MAX_EXTENTS	-	NUMBER	0
PCT_INCREASE	-	NUMBER	0
FREELISTS	-	NUMBER	0
FREELIST_GROUPS	-	NUMBER	0
LOGGING	-	VARCHAR2(1)	NULL
BACKED_UP	-	VARCHAR2(1)	NULL
NUM_ROWS	-	NUMBER	0
BLOCKS	-	NUMBER	0
EMPTY_BLOCKS	-	NUMBER	0
AVG_SPACE	-	NUMBER	0
CHAIN_CNT	-	NUMBER	0
AVG_ROW_LEN	-	NUMBER	0
AVG_SPACE_FREELIST_BLOCKS	-	NUMBER	0
NUM_FREELIST_BLOCKS	-	NUMBER	0
DEGREE	-	VARCHAR2(1)	NULL
INSTANCES	-	VARCHAR2(1)	NULL

Table C–19 (Cont.) DBA_TABLES

Name	Null?	Type	Value
CACHE	-	VARCHAR2(1)	NULL
TABLE_LOCK	-	VARCHAR2(1)	NULL
SAMPLE_SIZE	-	NUMBER	0
LAST_ANALYZED	-	DATE	NULL
PARTITIONED	-	VARCHAR2(1)	NULL
IOT_TYPE	-	VARCHAR2(1)	NULL
TEMPORARY	-	VARHCAR2(1)	NULL
SECONDARY	-	VARCHAR2(1)	NULL
NESTED	-	VARCHAR2(1)	NULL
BUFFER_POOL	-	VARCHAR2(1)	NULL
ROW_MOVEMENT	-	VARCHAR2(1)	NULL
GLOBAL_STATS	-	VARCHAR2(1)	NULL
USER_STATS	-	VARCHAR2(1)	NULL
DURATION	-	VARHCAR2(1)	NULL
SKIP_CORRUPT	-	VARCHAR2(1)	NULL
MONITORING	-	VARCHAR2(1)	NULL

Table C–20 DICT_COLUMNS

Name	Null?	Type	Value
TABLE_NAME	-	VARCHAR2(256)	-
COLUMN_NAME	-	VARCHAR2(256)	-
COMMENTS	-	VARCHAR2(1)	NULL

Table C–21 DICTIONARY

Name	Null?	Type	Value
TABLE_NAME	-	VARCHAR2(256)	-
COMMENTS	-	VARCHAR2(1)	-

Table C–22 DUAL

Name	Null?	Type	Value
DUMMY	NOT NULL	VARCHAR2(1)	"X"

Table C–23 TABLE_PRIVILEGES

Name	Null?	Type	Value
GRANTEE	-	VARCHAR2(256)	-
OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
GRANTOR	-	VARCHAR2(256)	-
SELECT_PRIV	-	VARCHAR2(1)	"Y"
INSERT_PRIV	-	VARCHAR2(1)	"A"
DELETE_PRIV	-	VARCHAR2(1)	"Y"
UPDATE_PRIV	-	VARCHAR2(1)	"A"
REFERENCES_PRIV	-	VARCHAR2(1)	"A"
ALTER_PRIV	-	VARCHAR2(1)	"Y"
INDEX_PRIV	-	VARCHAR2(1)	"Y"
CREATED	NOT NULL	DATE	-

Table C–24 USER_CATALOG

Name	Null?	Type	Value
TABLE_NAME	-	VARCHAR2(256)	-
TABLE_TYPE	-	VARCHAR2(5)	"TABLE" or "VIEW"

Table C–25 USER_COL_COMMENTS

Name	Null?	Type	Value
TABLE_NAME	-	VARCHAR2(256)	-
COLUMN_NAME	-	VARCHAR2(256)	-
COMMENTS	-	VARCHAR2(1)	NULL

Table C–26 USER_CONS_COLUMNS

Name	Null?	Type	Value
OWNER	NOT NULL	VARCHAR2(30)	-
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)	-
TABLE_NAME	NOT NULL	VARCHAR2(30)	-
COLUMN_NAME	-	VARCHAR2(8192)	-
POSITION	-	FLOAT(53)	-

Table C–27 USER_CONSTRAINTS

Name	Null?	Type	Value
OWNER	-	VARCHAR2(256)	-
CONSTRAINT_NAME	-	VARCHAR2(256)	-
CONSTRAINT_TYPE	-	VARCHAR2(1)	"R" or "P" or "U" or "C"
TABLE_NAME	-	VARCHAR2(256)	-
SEARCH_CONDITION	-	VARCHAR2(1)	NULL
R_OWNER	-	VARCHAR2(256)	-
R_CONSTRAINT_NAME	-	VARCHAR2(256)	-
DELETE_RULE	-	VARCHAR2(1)	NULL
STATUS	-	VARCHAR2(1)	NULL
DEFERRABLE	-	VARCHAR2(1)	NULL
DEFERRED	-	VARCHAR2(1)	NULL
VALIDATED	-	VARCHAR2(1)	NULL
GENERATED	-	VARCHAR2(1)	NULL
BAD	-	VARCHAR2(1)	NULL
RELY	-	VARCHAR2(1)	NULL
LAST_CHANGE	-	DATE	-

Table C–28 USER_IND_COLUMNS

Name	Null?	Type	Value
INDEX_NAME	NOT NULL	VARCHAR2(30)	-
TABLE_NAME	NOT NULL	VARCHAR2(30)	-
COLUMN_NAME	-	VARCHAR2(8192)	-
COLUMN_POSITION	NOT NULL	FLOAT(53)	-
COLUMN_LENGTH	NOT NULL	FLOAT(53)	-
DESCEND	-	VARCHAR2(4)	-

Table C–29 USER_INDEXES

Name	Null?	Type	Value
INDEX_NAME	-	VARCHAR2(256)	-
INDEX_TYPE	-	VARCHAR2(1)	NULL
TABLE_OWNER	-	VARCHAR2(256)	-
TABLE_NAME	-	VARCHAR2(256)	-
TABLE_TYPE	-	VARCHAR2(7)	"TABLE" or "CLUSTER"

Table C–29 (Cont.) USER_INDEXES

Name	Null?	Type	Value
UNIQUENESS	-	VARCHAR2(1)	NULL
COMPRESSION	-	VARCHAR2(1)	NULL
PREFIX_LENGTH	-	NUMBER	0
TABLESPACE_NAME	-	VARCHAR2(1)	NULL
INI_TRANS	-	NUMBER	0
MAX_TRANS	-	NUMBER	0
INITIAL_EXTENT	-	NUMBER	0
NEXT_EXTENT	-	NUMBER	0
MIN_EXTENTS	-	NUMBER	0
MAX_EXTENTS	-	NUMBER	0
PCT_INCREASE	-	NUMBER	0
PCT_THRESHOLD	-	NUMBER	0
INCLUDE_COLUMN	-	NUMBER	0
FREELISTS	-	NUMBER	0
FREELIST_GROUPS	-	NUMBER	0
PCT_FREE	-	NUMBER	0
LOGGING	-	VARCHAR2(1)	NULL
BLEVEL	-	NUMBER	0
LEAF_BLOCKS	-	NUMBER	0
DISTINCT_KEYS	-	NUMBER	0
AVG_LEAF_BLOCKS_PER_KEY	-	NUMBER	0
AVG_DATA_BLOCKS_PER_KEY	-	NUMBER	0
CLUSTERING_FACTOR	-	NUMBER	0
STATUS	-	VARCHAR2(1)	NULL
NUM_ROWS	-	NUMBER	0
SAMPLE_SIZE	-	NUMBER	0
LAST_ANALYZED	-	DATE	NULL
DEGREE	-	VARCHAR2(1)	NULL
INSTANCES	-	VARCHAR2(1)	NULL
PARTITIONED	-	VARCHAR2(1)	NULL
TEMPORARY	-	VARCHAR2(1)	NULL
GENERATED	-	VARCHAR2(1)	NULL
SECONDARY	-	VARCHAR2(1)	NULL
BUFFER_POOL	-	VARCHAR2(1)	NULL
USER_STATS	-	VARCHAR2(1)	NULL
DURATION	-	VARHCAR2(1)	NULL
PCT_DIRECT_ACCESS	-	NUMBER	0

Table C–29 (Cont.) USER_INDEXES

Name	Null?	Type	Value
ITYP_OWNER	-	VARCHAR2(1)	NULL
ITYP_NAME	-	VARCHAR2(1)	NULL
PARAMETERS	-	VARCHAR2(1)	NULL
GLOBAL_STATS	-	VARCHAR2(1)	NULL
DOMIDX_STATUS	-	VARCHAR2(1)	NULL
DOMIDX_OPSTATUS	-	VARCHAR2(1)	NULL
FUNCIDX_STATUS	-	VARCHAR2(1)	NULL

Table C–30 USER_OBJECTS

Name	Null?	Type	Value
OBJECT_NAME	-	VARCHAR2(256)	-
SUBOBJECT_NAME	-	VARCHAR2(1)	NULL
OBJECT_ID	-	NUMBER	-
DATA_OBJECT_ID	-	NUMBER	0
OBJECT_TYPE	-	VARCHAR2(9)	"TABLE" or "VIEW" or "INDEX" or "PROCEDURE"
CREATED	-	DATE	-
LAST_DDL_TIME	-	DATE	-
TIMESTAMP	-	VARCHAR2(1)	NULL
STATUS	-	VARCHAR2(5)	"VALID"
TEMPORARY	-	VARCHAR2(1)	NULL
GENERATED	-	VARCHAR2(1)	NULL
SECONDARY	-	VARCHAR2(1)	NULL

Table C–31 USER_TAB_COLUMNS

Name	Null?	Type	Value
TABLE_NAME	-	VARCHAR2(256)	-
COLUMN_NAME	-	VARCHAR2(256)	-
DATA_TYPE	-	VARCHAR2(8)	-
DATA_TYPE_MOD	-	VARCHAR2(1)	NULL
DATA_TYPE_OWNER	-	VARCHAR2(1)	NULL
DATA_LENGTH	-	NUMBER	-
DATA_PRECISION	-	NUMBER	-
DATA_SCALE	-	NUMBER	-

Table C–31 (Cont.) USER_TAB_COLUMNS

Name	Null?	Type	Value
NULLABLE	-	VARCHAR2(1)	"Y" or "N"
COLUMN_ID	-	NUMBER	-
DEFAULT_LENGTH	-	NUMBER	0
DATA_DEFAULT	-	VARCHAR2(1)	NULL
NUM_DISTINCT	-	NUMBER	0
LOW_VALUE	-	NUMBER	0
HIGH_VALUE	-	NUMBER	0
DENSITY	-	NUMBER	0
NUM_NULLS	-	NUMBER	0
NUM_BUCKETS	-	NUMBER	0
LAST_ANALYZED	-	DATE	NULL
SAMPLE_SIZE	-	NUMBER	0
CHARACTER_SET_NAME	-	VARCHAR2(1)	NULL
CHAR_COL_DECL_LENGTH	-	NUMBER	0
GLOBAL_STATS	-	VARCHAR2(1)	NULL
USER_STATS	-	VARCHAR2(1)	NULL
AVG_COL_LEN	-	NUMBER	0

Table C–32 USER_TAB_COMMENTS

Name	Null?	Type	Value
TABLE_NAME	-	VARCHAR2(256)	-
TABLE_TYPE	-	VARCHAR2(5)	"TABLE" or "VIEW"
COMMENTS	-	VARCHAR2(1)	NULL

Table C–33 USER_TABLES

Name	Null?	Type	Value
TABLE_NAME	-	VARCHAR2(256)	-
TABLESPACE_NAME	-	VARCHAR2(1)	NULL
CLUSTER_NAME	-	VARCHAR2(1)	NULL
IOT_NAME	-	VARCHAR2(1)	NULL
PCT_FREE	-	NUMBER	0
PCT_USED	-	NUMBER	0
INI_TRANS	-	NUMBER	0
MAX_TRANS	-	NUMBER	0

Table C–33 (Cont.) USER_TABLES

Name	Null?	Type	Value
INITIAL_EXTENT	-	NUMBER	0
NEXT_EXTENT	-	NUMBER	0
MIN_EXTENTS	-	NUMBER	0
MAX_EXTENTS	-	NUMBER	0
PCT_INCREASE	-	NUMBER	0
FREELISTS	-	NUMBER	0
FREELIST_GROUPS	-	NUMBER	0
LOGGING	-	VARCHAR2(1)	NULL
BACKED_UP	-	VARCHAR2(1)	NULL
NUM_ROWS	-	NUMBER	0
BLOCKS	-	NUMBER	0
EMPTY_BLOCKS	-	NUMBER	0
AVG_SPACE	-	NUMBER	0
CHAIN_CNT	-	NUMBER	0
AVG_ROW_LEN	-	NUMBER	0
AVG_SPACE_FREELIST_BLOCKS	-	NUMBER	0
NUM_FREELIST_BLOCKS	-	NUMBER	0
DEGREE	-	VARCHAR2(1)	NULL
INSTANCES	-	VARCHAR2(1)	NULL
CACHE	-	VARCHAR2(1)	NULL
TABLE_LOCK	-	VARCHAR2(1)	NULL
SAMPLE_SIZE	-	NUMBER	0
LAST_ANALYZED	-	DATE	NULL
PARTITIONED	-	VARCHAR2(1)	NULL
IOT_TYPE	-	VARCHAR2(1)	NULL
TEMPORARY	-	VARHCAR2(1)	NULL
SECONDARY	-	VARCHAR2(1)	NULL
NESTED	-	VARCHAR2(1)	NULL
BUFFER_POOL	-	VARCHAR2(1)	NULL
ROW_MOVEMENT	-	VARCHAR2(1)	NULL
GLOBAL_STATS	-	VARCHAR2(1)	NULL
USER_STATS	-	VARCHAR2(1)	NULL
DURATION	-	VARCHAR2(1)	NULL
SKIP_CORRUPT	-	VARCHAR2(1)	NULL
MONITORING	-	VARCHAR2(1)	NULL

Table C–34 USER_USERS

Name	Null?	Type	Value
USERNAME	-	VARCHAR2(256)	-
USER_ID	NOT NULL	NUMBER(5)	-
ACCOUNT_STATUS	-	VARCHAR2(4)	"OPEN"
LOCK_DATE	-	DATE	NULL
EXPIRY_DATE	-	DATE	NULL
DEFAULT_TABLESPACE	-	VARCHAR2(1)	NULL
TEMPORARY_TABLESPACE	-	VARCHAR2(1)	NULL
CREATED	NOT NULL	DATE	-
INITIAL_RSRC_CONSUMER_GROUP	-	VARCHAR2(1)	NULL
EXTERNAL_NAME	-	VARCHAR2(1)	NULL

Table C–35 USER_VIEWS

Name	Null?	Type	Value
VIEW_NAME	-	VARCHAR2(256)	-
TEXT_LENGTH	-	NUMBER	0
TEXT	-	VARCHAR2(256)	-
TYPE_TEXT_LENGTH	-	NUMBER	0
TYPE_TEXT	-	VARCHAR2(1)	NULL
OID_TEXT_LENGTH	-	NUMBER	0
OID_TEXT	-	VARCHAR2(1)	NULL
VIEW_TYPE_OWNER	-	VARCHAR2(1)	NULL
VIEW_TYPE	-	VARCHAR2(1)	NULL

D

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.

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 user name 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 SQL Server Initialization Parameters

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

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_LONG_PIECE_TRANSFER_SIZE

Property	Description
Default value	64 KB
Range of values	Any value up to 2 GB

Sets the size of the piece of LONG data being transferred. A smaller piece size means less memory requirement, but more round-trips to fetch all the data. A larger piece size means fewer round-trips, but more of a memory requirement to store the intermediate pieces internally. Thus, the initialization parameter can be used to tune a system for the best performance, with the best trade-off between round-trips and memory requirements, and network latency or response time.

HS_OPEN_CURSORS

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

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 to 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_TIME_ZONE

Property	Description
Default value for '[+|-]hh:mm'	Derived from the NLS_TERRITORY initialization parameter
Range of values for '[+|-]hh:mm'	Any valid datetime format mask

Specifies the default local time zone displacement for the current SQL session. The format mask, [+|-]hh:mm, is specified to indicate the hours and minutes before or after UTC (Coordinated Universal Time—formerly Greenwich Mean Time). For example:

```
HS_TIME_ZONE = [+ | -] hh:mm
```

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.
- 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.

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*

HS_FDS_CONNECT_INFO

Property	Description
Default Value	None
Range of Values	Not applicable

HS_FDS_CONNECT_INFO which describes the connection to the non-Oracle system.

The default initialization parameter file already has an entry for this parameter. The syntax for HS_FDS_CONNECT_INFO for the gateway is as follows:

For UNIX:

```
HS_FDS_CONNECT_INFO=host_name[[:port_number]|/[instance_name]][/database_name]
```

where, *host_name* is the host name or IP address of the machine hosting the SQL Server database, *port_number* is the port number of the SQL Server database server, *instance_name* is the instance of SQL Server running on the machine, and *database_name* is the SQL Server database name.

Either of the variables *port_number* or *instance_name* can be used, but not both together. Optionally, they both can be omitted. The variable *database_name* is always optional. The slash (/) is required when a particular value is omitted. For example, all of the following entries are valid:

```
HS_FDS_CONNECT_INFO=host_name/instance_name/database_name
HS_FDS_CONNECT_INFO=host_name//database_name
HS_FDS_CONNECT_INFO=host_name:port_name//database_name
```

```
HS_FDS_CONNECT_INFO=host_name/instance_name
HS_FDS_CONNECT_INFO=host_name
```

For Windows:

```
HS_FDS_CONNECT_INFO= host_name/[instance_name][/database_name]
```

where, *host_name* is the host name or IP address of the machine hosting the SQL Server database, *instance_name* is the instance of SQL Server running on the machine, and *database_name* is the SQL Server database name.

Both *instance_name* and *database_name* are optional. If *instance_name* is omitted and *database_name* is provided, the slash (/) is required. This can be shown as follows:

```
HS_FDS_CONNECT_INFO= host_name//database_name
```

HS_FDS_DEFAULT_OWNER

Property	Description
Default Value	None
Range of Values	Not applicable

The name of the table owner that is used for the non-Oracle database tables if an owner is not specified in the SQL statements.

> **Note:** If this parameter is not specified and the owner is not explicitly specified in the SQL statement, then the user name of the Oracle user or the user name specified when creating the database link is used.

HS_FDS_PROC_IS_FUNC

Property	Description
Default Value	FALSE
Range of Values	TRUE, FALSE

Enables return values from functions. By default, all stored procedures and functions do not return a return value to the user.

> **Note:** If you set this initialization parameter, you must change the syntax of the procedure execute statement for all existing stored procedures to handle return values.

HS_FDS_RECOVERY_ACCOUNT

Property	Description
Default Value	RECOVER

Property	Description
Range of values	Any valid user ID

Specifies the name of the recovery account used for the commit-confirm transaction model. An account with user name and password must be set up at the non-Oracle system. For more information about the commit-confirm model, see the HS_TRANSACTION_MODEL parameter.

The name of the recovery account is case sensitive.

HS_FDS_RECOVERY_PWD

Property	Description
Default Value	RECOVER
Range of values	Any valid password

Specifies the password of the recovery account used for the commit-confirm transaction model set up at the non-Oracle system. For more information about the commit-confirm model, see the HS_TRANSACTION_MODEL parameter.

The name of the password of the recovery account is case sensitive.

HS_FDS_RESULTSET_SUPPORT

Property	Description
Default Value	FALSE
Range of Values	TRUE, FALSE

Enables result sets to be returned from stored procedures. By default, all stored procedures do not return a result set to the user.

Note: If you set this initialization parameter, you must do the following:

- Change the syntax of the procedure execute statement for all existing stored procedures, to handle result sets
- Work in the sequential mode of Heterogeneous Services

HS_FDS_TRACE_LEVEL

Property	Description
Default Value	OFF
Range of values	OFF, ON, DEBUG

Specifies whether error tracing is turned on or off for gateway connectivity.

The following values are valid:

- OFF disables the tracing of error messages.
- ON enables the tracing of error messages that occur when you encounter problems. The results are written by default to a gateway log file in LOG directory where the gateway is installed.
- DEBUG enables the tracing of detailed error messages that can be used for debugging.

HS_FDS_TRANSACTION_LOG

Property	Description
Default Value	HS_TRANSACTION_LOG
Range of Values	Any valid table name

Specifies the name of the table created in the non-Oracle system for logging transactions. For more information about the transaction model, see the HS_TRANSACTION_MODEL parameter.

HS_FDS_REPORT_REAL_AS_DOUBLE

Property	Description
Default Value	FALSE
Range of Values	TRUE, FALSE

Enables Oracle Database Gateway for SQL Server to treat SINGLE FLOAT PRECISION fields as DOUBLE FLOAT PPRECISION fields.

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.

Index

H

I

K

L

M

N

O

P

R

S

T

U

V

W