Oracle® Provider for OLE DB

Developer’s Guide

Release 9.2

March 2002

Part No. A95498-01

Oracle Provider for OLE DB Developer’s Guide, Release 9.2

Part No. A95498-01

Contributors: Janis Greenberg, Eric Belden, Riaz Ahmed, Kiminari Akiyama, Neeraj Gupta, Sinclair Hsu, Gopal Kirsur, Sunil Mushran, Rajendra Pingte, Helen Slattery

Contents

2 Features of OraOLEDB

Send Us Your Comments

Oracle Provider for OLE DB Developer's Guide, Release 9.2

Part No. A95498-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: ntdoc_us@oracle.com
- FAX: (650) 506-7365 Attn: Oracle Database for Windows Documentation
- Postal service:
  Oracle Corporation
  Oracle Database for Windows Documentation Manager
  500 Oracle Parkway, Mailstop 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Based on an open standard, Oracle Provider for OLE DB (OraOLEDB) allows access to Oracle databases. This documentation describes OraOLEDB's provider-specific features and properties.

This manual describes only the features of Oracle9*i* for Windows software that apply to the Windows NT, Windows 2000, Windows XP, and Windows 98 operating systems. Information on Oracle9*i* Personal Edition software on Windows 98 is not covered in this manual.

This preface contains these topics:

Audience

Oracle Provider for OLE DB Developer's Guide is intended for programmers developing applications to access an Oracle database using Oracle Provider for OLE DB. This documentation is also valuable to systems analysts, project managers, and others interested in the development of database applications.

To use this document, you must be familiar with OLE DB and have a working knowledge of application programming using Microsoft C/C++, Visual Basic, or ActiveX Data Objects (ADO). Knowledge of Component Object Model (COM) concepts are also useful.

Readers should also be familiar with the use of Structured Query Language (SQL) to access information in relational database systems.

Organization

This document contains:

Chapter 1, "Introduction to Oracle Provider for OLE DB"

This chapter discusses OLE DB, Oracle Provider for OLE DB (OraOLEDB), requirements, and installation.

Chapter 2, "Features of OraOLEDB"

This chapter discusses OraOLEDB components and describes how to use OraOLEDB to develop consumer applications.

Appendix A, "Provider-Specific Information"

This appendix discusses OLE DB information that is specific to Oracle Provider for OLE DB.

Glossary

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Database Installation Guide for Windows*
- *Oracle9i Database Release Notes for Windows*
- *Oracle9i Database Administrator's Guide for Windows*

- *Oracle Services for Microsoft Transaction Server Developer's Guide*
- *Oracle Enterprise Manager Administrator's Guide*
- *Oracle9i Net Services Administrator's Guide*
- *Oracle9i Database New Features*
- *Oracle9i Database Concepts*
- *Oracle9i Database Reference*
- *Oracle9i Database Error Messages*
- *Oracle9i Database Globalization Support Guide*

In North America, printed documentation is available for sale in the Oracle Store at

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

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

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/admin/account/membership.html`

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

`http://otn.oracle.com/docs/index.htm`

To access the database documentation search engine directly, please visit

`http://tahiti.oracle.com`

For additional information, see:

`www.microsoft.com/data/oledb/`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.

Convention	Meaning	Example
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true.` Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`

|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}`
		`[COMPRESS | NOCOMPRESS]`

Convention	Meaning	Example

`. . .`	Horizontal ellipsis points indicate either:
	■ That we have omitted parts of the code that are not directly related to the example
	■ That you can repeat a portion of the code
		`CREATE TABLE ... AS` *`subquery`*`;`
		
		`SELECT` *`col1`*`,` *`col2`*`, ... ,` *`coln`* `FROM`
		`employees;`

`.`
`.`
`.`
	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	`SQL> SELECT NAME FROM V$DATAFILE;`
		`NAME`
		`------------------------------------`
		`/fsl/dbs/tbs_01.dbf`
		`/fs1/dbs/tbs_02.dbf`
		`.`
		`.`
		`.`
		`/fsl/dbs/tbs_09.dbf`
		`9 rows selected.`

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);`
		`acct    CONSTANT NUMBER(4) := 3;`

`Italics`	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`*
		`DB_NAME =` *`database_name`*

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM`
		`employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM`
		`employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose Start >	How to start a program. For example, to start Database Configuration Assistant, you must click the Start button on the taskbar and then choose Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant.	Choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant
File and Directory Names	File and directory names are not case sensitive. The special characters <, >, :, ", /, |, and - are not allowed. The special character \ is treated as an element separator, even when it appears in quotes. If the file name begins with \\, Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32` is the same as `C:\WINNT\SYSTEM32`
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is "^". Your prompt reflects the subdirectory in which you are working. Referred to as the command prompt in this manual.	`C:\oracle\oradata>`

Special characters	The backslash special character (\) is sometimes required as an escape character for the double quote (") special character at the Windows command prompt. Parentheses and the single quote special character (') do not require an escape character. See your Windows operating system documentation for more information on escape and special characters.	`C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\"`
		
		`C:\>imp SYSTEM/`*`password`*` FROMUSER=scott TABLES=(emp, dept)`

HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default was: ▪ `C:\orant` for Windows NT ▪ `C:\orawin98` for Windows 98 or whatever you called your Oracle home. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\ora`*nn* where *nn* is the latest release number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this manual follow OFA conventions. See *Oracle9i Database Getting Started for Windows* for additional information on OFA compliance and for information on installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

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. Standards will continue to evolve over time, and Oracle Corporation 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 additional information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, 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, JAWS 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 Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

What's New in OraOLEDB?

The following sections describe the new features in OraOLEDB:

- New Features in Oracle Provider for OLE DB for Release 9.2
- New Features in Oracle Provider for OLE DB for Release 9.0.1
- New Features in Oracle Provider for OLE DB for Release 8.1.7

New Features in Oracle Provider for OLE DB for Release 9.2

This section contains these topics:

- **Support for using OraOLEDB with OLE DB .NET Data Provider**

  ADO.NET applications developers can use OraOLEDB through the OLE DB .NET Data Provider. A connection attribute, `OLEDB.NET`, can be set at connection time for OraOLEDB to be compatible with OLE DB .NET Data Provider. See "OLE DB .NET Data Provider Compatibility" on page 2-24

New Features in Oracle Provider for OLE DB for Release 9.0.1

This section contains these topics:

- **Using Oracle9*i* on Windows 2000**

  There are some differences between using Oracle9*i* on Windows 2000 and Windows NT 4.0.

  See Also: *Oracle9i Database Getting Started for Windows*

New Features in Oracle Provider for OLE DB for Release 8.1.7

Oracle8*i* release 8.1.7 included the following:

- **Support for returning multiple rowsets.**

  Consumers can use this feature to access all the `REF CURSOR`s returned by a stored procedure. See "Multiple Rowsets" on page 2-12.

- **Support for the Unicode character set.**

  Using this feature, consumers can use OraOLEDB to access data in multiple languages on the same client computer. It can be especially useful in creating global Internet applications supporting as many languages as the Unicode standard entails. For example, one can write a single Active Server Page (ASP) that accesses an Oracle8*i* database to dynamically generate content in Japanese, Arabic, English, and Thai. See "Unicode Support" on page 2-22 and "Datatype Mappings in Rowsets and Parameters" on page A-2.

1 Introduction to Oracle Provider for OLE DB

This chapter introduces Oracle Provider for OLE DB (OraOLEDB).

This chapter contains these topics:

- Overview of OLE DB
- Overview of OraOLEDB
- System Requirements
- OraOLEDB Installation

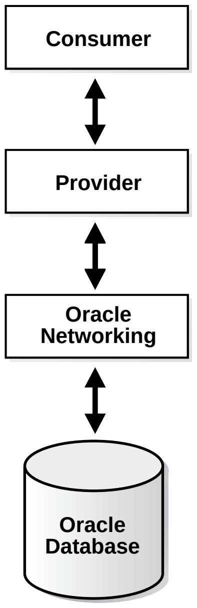
Overview of OLE DB

OLE DB is an open standard data access methodology which utilizes a set of **Component Object Model (COM)** interfaces for accessing and manipulating different types of data. These interfaces are available from various database providers.

OLE DB Design

OLE DB's design centers around the concept of a **consumer** and **provider**. Figure 1–1, "OLE DB Flow" is an illustration of the OLE DB system. The consumer represents the traditional client. The provider places data into a tabular format and returns it to the consumer.

Figure 1–1 OLE DB Flow



OLE DB Data Providers

OLE DB data providers are a set of **COM** components that transfer data from a data source to a **consumer**. An OLE DB Provider places that data in a tabular format in response to calls from a consumer. Providers can be simple or complex. A **provider** may return a table, it may allow the consumer to determine the format of that table, and it may perform operations on the data.

Each provider implements a standard set of COM interfaces to handle requests from the consumer. A provider may implement optional COM interfaces to provide additional functionality.

With the standard interfaces, any OLE DB consumer can access data from any provider. Because of COM components, consumers can access them in any programming language that supports COM, such as C++, Visual Basic, and Java.

OLE DB Data Consumers

The OLE DB data consumer is any application or tool that utilizes OLE DB interfaces of a provider to access a broad range of data.

Overview of OraOLEDB

Oracle Provider for OLE DB (OraOLEDB) is an OLE DB data provider that offers high performance and efficient access to Oracle data by OLE DB consumers.

In general, this developer’s guide assumes that you are using OraOLEDB through OLE DB or ADO.

For sample code, the latest patches, and other technical information on the Oracle Provider for OLE DB, go to

```
http://otn.oracle.com/tech/windows/ole_db
```

With the advent of the .NET framework, support has been provided for using the OLE DB .NET Data Provider with OraOLEDB. With the proper connection attribute setting, an OLE DB .NET Data Provider can utilize OraOLEDB to access Oracle.

> **See Also:** "OLE DB .NET Data Provider Compatibility" on page 2-24 for further information on support for OLE DB .NET Data Provider

System Requirements

The following items are required on a system to use Oracle Provider for OLE DB:

- Windows 98, Windows NT 4.0, Windows 2000, or Windows XP
- Access to an Oracle Server (release 8 or later)
- **Oracle Net Services**
- Redistributable files provided with Microsoft Data Access Components (MDAC) 2.1 or higher are required by the provider. These files are available at the Microsoft Web site:

  `www.microsoft.com/data/oledb/`

- Oracle Services for Microsoft Transaction Server (release 9.0). This item is required for consumers using Microsoft Transaction Server (MTS) or COM+.

Note: With the Oracle Services for Microsoft Transaction Server installed, OraOLEDB supports MTS against database versions Oracle9*i* release 1 (9.0.1), Oracle8*i* (8.1.5 or higher) and Oracle8 (8.0.6 or higher).

OraOLEDB Installation

Oracle Provider for OLE DB is included as part of your Oracle installation. It contains the features and demos that illustrate how to use this product to solve real-world problems.

See Also: The *Oracle9i Database Installation Guide for Windows* for installation instructions

During the installation process, the files listed in Table 1–1 are installed on the system.

Table 1–1 Oracle Provider for OLE DB Files

File	Description	Location
`OraOLEDB.dll`	Oracle Provider for OLE DB	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDBrfc.dll`	Oracle rowset file cache manager	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDBrmc.dll`	Oracle rowset memory cache manager	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDBrst.dll`	Oracle rowset	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDBgmr.dll`	Oracle ODBC SQL parser	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDB`*lang*`.dll` where *lang* is the appropriate language	Language-specific resource DLL	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDBpus.dll`	Property descriptions	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDButl.dll`	OraOLEDB utility DLL	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDB.tlb`	OraOLEDB type library	*ORACLE_BASE*\ORACLE_HOME\bin
`OraOLEDB.h`	OraOLEDB header file	*ORACLE_BASE*\ORACLE_HOME\oledb\include
`OraOLEDB.lib`	OraOLEDB library file	*ORACLE_BASE*\ORACLE_HOME\oledb\lib

Table 1–1 Oracle Provider for OLE DB Files

File	Description	Location
`OraOLEDB`*`lang`*`.msb` where *`lang`* is the appropriate language	Language-specific message file	*`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\oledb\mesg`
readme and documentation files	Release notes and online documentation	*`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\oledb\doc`
sample files	Sample code	*`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\oledb\samples`

2 Features of OraOLEDB

This chapter describes components of Oracle Provider for OLE DB (OraOLEDB) and how to use the components to develop OLE DB consumer applications.

This chapter contains these topics:

- OraOLEDB Provider Specific Features
- Using OraOLEDB with Visual Basic

OraOLEDB Provider Specific Features

The following sections describe provider-specific features of OraOLEDB:

- Data Source
- OraOLEDB Sessions
- Commands
- Rowsets
- LOB Support
- Unicode Support
- Using OraOLEDB with Visual Basic
- OLE DB .NET Data Provider Compatibility

Additional provider-specific information is provided in Appendix A, "Provider-Specific Information".

Data Source

A data source object in OraOLEDB is responsible for establishing the first connection to the Oracle database. To establish the initial connection, the consumer must use the `CoCreateInstance` function to create an instance of the data source object. This function requires important information about the provider: class ID of the provider and executable context. The class ID of OraOLEDB is `CLSID_OraOLEDB`.

OraOLEDB is an in-process server. When calling `CoCreateInstance`, use the `CLSCTX_INPROC_SERVER` macro. For example:

```
// create an instance of OraOLEDB data source object and
// obtain the IDBInitialize interface
hr = CoCreateInstance(CLSID_OraOLEDB, NULL,
                      CLSCTX_INPROC_SERVER, IID_IDBInitialize,
                     (void**)&pIDBInitialize);
```

Note: OraOLEDB does not support persistent data source objects.

After the successful creation of an instance of a data source object, the consumer application can initialize the data source and create sessions.

OraOLEDB supports connections to Oracle databases release 7.3.4 and higher. To connect to a specific database, the consumer is required to set the following properties of the `DBPROPSET_DBINIT` property set:

- `DBPROP_AUTH_USERNAME` with the user ID, such as `scott`
- `DBPROP_AUTH_PASSWORD` with the password, such as `tiger`
- `DBPROP_INIT_DATASOURCE` with the net service name, such as `myOraDb`

The consumer could also populate `DBPROP_INIT_PROMPT` with `DBPROMPT_PROMPT` which causes the **provider** to display a logon box for the user to enter the connect information.

Using `DBPROMPT_NOPROMPT` disables display of the logon box. In this case, incomplete logon information causes the provider to return a logon error. However, if this property is set to `DBPROMPT_COMPLETE` or `DBPROMPT_COMPLETEREQUIRED`, the logon box will only be displayed if the logon information is incomplete.

Connecting to an Oracle Database

To connect to an Oracle database using OraOLEDB, the OLE DB connection string must be as follows:

```
"Provider=OraOLEDB.Oracle;User ID=user;Password=pwd;Data Source=constr;"
```

When connecting to a remote database, `Data Source` must be set to the appropriate net service name which is the alias in the `tnsnames.ora` file. For more information, refer to *Oracle9i Net Services Administrator's Guide*.

OraOLEDB-specific Connection String Attributes

OraOLEDB offers provider-specific Connection String attributes, which are set in the same way as the Provider and User ID are set. The provider-specific connection string attributes are:

- `CacheType` - specifies the type of cache used to store the rowset data on the client. See "OraOLEDB-specific Connection String Attributes for Rowsets" on page 2-18.
- `ChunkSize` - specifies the size of `LONG` or `LONG RAW` column data stored in the provider's cache. See "OraOLEDB-specific Connection String Attributes for Rowsets" on page 2-18.
- `DistribTX` - enables or disables distributed transaction enlistment capability. See "Distributed Transactions" on page 2-4.

- `FetchSize` - specifies the size of the fetch array in rows. See "OraOLEDB-specific Connection String Attributes for Rowsets" on page 2-18.
- `OSAuthent` - specifies whether OS Authentication will be used when connecting to an Oracle database. See "OS Authentication" on page 2-4.
- `PLSQLRSet` - enables or disables the return of a rowset from **PL/SQL** stored procedures. See "OraOLEDB Custom Properties for Commands" on page 2-8.
- `PwdChgDlg` - enables or disables displaying the password change dialog box when the password expires. See "Password Expiration" on page 2-5.
- `OLEDB.NET` - enables or disables compatibility with OLE DB .NET Data Provider. See "OLE DB .NET Data Provider Compatibility" on page 2-24.

Default Attribute Values

The default values for these attributes are located under the `\\HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\OLEDB` registry key.

The registry default values are read by OraOLEDB from the registry when the provider is loaded into memory. If Oracle-specific connection string attributes are not provided at connection time, the default registry values are used. However, if the attributes are provided, these new values override the default registry values.

These attributes can be set by setting the `DBPROP_INIT_PROVIDERSTRING` property, provided in the `DBPROPSET_DBINIT` property set. For example:

```
"FetchSize=100;CacheType=Memory;OSAuthent=0;PLSQLRSet=1;"
```

Distributed Transactions

The `DistribTX` attribute specifies whether sessions are enabled to enlist in distributed transactions. Valid values are `0` (disabled) and `1` (enabled). The default is `1` which indicates that sessions are enabled for distributed transaction enlistments.

Applications using Microsoft Transaction Server must have `DistribTX` set to `1`, the default.

OS Authentication

The `OSAuthent` attribute specifies whether OS authentication will be used when connecting to an Oracle database. Valid values are `0` (disabled) and `1` (enabled). The default is `0` which indicates that OS authentication is not used.

OS authentication is the feature by which Oracle uses the security mechanisms of the operating system to authorize users. For more information on this subject and how to set it up on Windows NT clients, refer to the information on authenticating database users on Windows NT in *Oracle9i Security and Network Integration Guide*.

After the Windows NT client has been set up properly for OS authentication, this feature may be enabled by OraOLEDB clients by setting any of the following:

- `DBPROP_AUTH_USERNAME` to "`/`"
- `DBPROP_INIT_PROVIDERSTRING` to "`OSAuthent=1`;"
- `OSAuthent` in the registry to "`1`"

Password Expiration

Oracle9*i* provides a Password Expiration feature which allows database administrators to force users to change their passwords regularly. The `PwdChgDlg` attribute enables or disables the displaying of the password change dialog, whenever a logon fails due to an expired password. When enabled, the provider displays the dialog to change the password. When disabled, the logon fails with an error message. The valid values are `0` (disabled) and `1` (enabled). The default is `1` (enabled). For more information on the Password Expiration feature, see *Oracle9i Database Administrator's Guide*.

Example: Connecting to an Oracle Database Using ADO

The following are examples illustrating how to connect to an Oracle database using OraOLEDB and ADO.

Note: If the Data Source, User ID, and Password are provided with the Open method, ADO ignores those `ConnectionString` attributes.

Connect using ConnectionString

```
Dim con As New ADODB.Connection
con.ConnectionString = "Provider=OraOLEDB.Oracle;Data Source=MyOraDb;" & _
                     "User ID=scott;Password=tiger;"
con.Open
```

Connect without using ConnectionString

```
Dim con As New ADODB.Connection
con.Provider = "OraOLEDB.Oracle"
con.Open "MyOraDb", "scott", "tiger"
```

Connect and set provider specific attributes

```
Dim con As New ADODB.Connection
con.Provider = "OraOLEDB.Oracle"
con.ConnectionString = "FetchSize=200;CacheType=Memory;" & _
                       "OSAuthent=0;PLSQLRSet=1;Data Source=MyOraDb;" & _
                       "User ID=scott;Password=tiger;"
con.Open
```

OS Authenticated connect setting user ID to "/"

```
Dim con As New ADODB.Connection
con.Provider = "OraOLEDB.Oracle"
con.Open "MyOraDb", "/", ""
```

OS Authenticated connect using OSAuthent

```
Dim con As New ADODB.Connection
con.Provider = "OraOLEDB.Oracle"
con.ConnectionString = "Data Source=MyOraDb;OSAuthent=1;"
con.Open
```

OraOLEDB Sessions

An OraOLEDB session object represents a single connection to an Oracle database. The session object exposes the interfaces that allow data access and manipulation.

The first session created on the initialized data source inherits the initial connection established by `IDBInitialize::Initialize()`. Subsequent sessions that are created establish their own independent connections to the particular Oracle server specified by the data source properties.

Each session object also defines a transaction space for a data source. All command and rowset objects created from a particular session object are part of the transaction of that session.

After all references to the session object are released, the session object is removed from memory and the connection is dropped.

Transactions

OraOLEDB supports local and distributed transactions which provide explicit commit and abort.

OraOLEDB does not support nested transactions. In addition, a local transaction cannot be started if the session is currently enlisted in a distributed transaction. This also applies to distributed transactions if the session is currently enlisted in a local transaction.

Local Transactions OraOLEDB supports the `ITransactionLocal` interface for explicit transactions. By default, OraOLEDB is in an autocommit mode, meaning that each unit of work done on the database is automatically or implicitly committed. With the use of `ITransactionLocal` interface, consumers may explicitly start a transaction for a particular session, allowing a unit of work to be explicitly committed or aborted by the consumer.

OraOLEDB supports the Read Committed (Cursor Stability) isolation level. In this level, the changes made by other transactions are not visible until those transactions are committed.

Distributed Transactions OraOLEDB consumers must install Oracle Services for Microsoft Transaction Server (MTS) release 9.0.1 or later to be able to participate in Microsoft Transaction Server (or COM+) transactions or to enlist in a distributed transaction coordinated by Microsoft Distributed Transaction Coordinator (MS DTC). For setup and configuration information on Oracle Services for MTS, see *Oracle Services for Microsoft Transaction Server Developer's Guide*.

OraOLEDB ignores `IsoLevel`, `IsoFlags`, and `pOtherOptions` parameters when `ITransactionJoin::JoinTransaction()` is called. These options must be provided when the consumer acquires a transaction object from MS DTC with the `ITransactionDispenser::BeginTransaction()` method call.

However, if `IsoFlags` is nonzero, `XACT_E_NOISORETAIN` is returned.

Commands

OraOLEDB supports ANSI SQL as supported by Oracle and the ODBC SQL syntax.

Stored Procedures

When executing an Oracle **PL/SQL** **stored procedure** using a command, use Oracle native syntax or the ODBC procedure call escape sequence in the command text:

- Oracle native syntax: `BEGIN credit_account(123, 40); END;`
- ODBC syntax: {`CALL credit_account(123, 40)`}

Preparing Commands

OraOLEDB validates and fetches the metadata only for SELECT SQL statements.

Command Parameters

When using Oracle ANSI SQL, parameters in the command text are preceded by a colon. In ODBC SQL, parameters are indicated by a question mark ("?").

OraOLEDB supports input, output, and input/output parameters for PL/SQL stored procedures and stored functions. OraOLEDB supports input parameters for SQL statements.

Note: OraOLEDB supports only positional binding.

OraOLEDB Custom Properties for Commands

OraOLEDB custom properties for Commands are grouped under the custom property set `ORAPROPSET_COMMANDS`. It provides these properties:

Table 2–1 Custom Properties for Commands

For Visual Basic Users	For C++ Users
`PLSQLRSet`	`ORAPROP_PLSQLRSet`
`NDatatype`	`ORAPROP_NDatatype`
`SPPrmsLOB`	`ORAPROP_SPPrmsLOB`

PLSQLRSet

This property is similar to the PLSQLRSet Connection string attribute.

The property specifies whether OraOLEDB must return a rowset from the PL/SQL stored procedure. If the stored procedure, provided by the consumer, returns a rowset, PLSQLRSet must be set to TRUE (enabled). This property should be set to FALSE after the command has been executed. By default, the property is set to FALSE (disabled).

Consumers should use the property over the attribute, as the property can be set at the command object rather than at the session. By setting it at the command object, the consumer is able to set the property only for the command object executing stored procedures which are returning rowsets. With the attribute, the consumer needed to set it even if only one of many stored procedures being executed by the ADO application returned a rowset. The use of this property should provide a performance boost to applications making use of the attribute previously.

Example: Setting the Custom Property PLSQLRSet

```
Dim objRes As NEW ADODB.Recordset
Dim objCon As NEW ADODB.Connection
Dim objCmd As NEW ADODB.Command
....
objCmd.ActiveConnection = objCon
objCmd.CommandType = adCmdText

’ Enabling the PLSQLRSet property indicates to the provider
’ that the command returns one or more rowsets
objCmd.Properties("PLSQLRSet") = TRUE

’ Assume Employees.GetEmpRecords() has a REF CURSOR as
’ one of the arguments
objCmd.CommandText = "{ CALL Employees.GetEmpRecords(?,?) }"

’ Execute the SQL
set objRes = objCmd.Execute

’ It is a good idea to disable the property after execute as the
’ same command object may be used for a different SQL statement
objCmd.Properties("PLSQLRSet") = FALSE
```

NDatatype

This property allows the consumers to specify whether any of the parameters bound to the command are of Oracle's N datatypes (`NCHAR`, `NVARCHAR` or `NCLOB`). This information is required by OraOLEDB to detect and bind the parameters appropriately. This property should not be set for commands executing `SELECT` statements. However, this property must be set for all other SQLs such as `INSERT`, `UPDATE`, and `DELETE`.

The use of this property should be limited to SQLs containing parameters of N datatype as setting it incurs a processing overhead of at least one round-trip to the database. By default, this property is set to `FALSE`.

Note: OraOLEDB does not support parameters of N datatypes in the `WHERE` clause of SQL statements.

Note: Consumers are required to use the ODBC procedure call escape sequence to call **stored procedure**s or functions having N datatype parameters.

Example: Setting the Custom Property NDatatype

```
Dim objCon As NEW ADODB.Connection
Dim objCmd As NEW ADODB.Command
Dim prEmpno As NEW ADODB.Parameter
Dim prEname As NEW ADODB.Parameter
...
objCmd.ActiveConnection = objCon
objCmd.CommandType = adCmdText

' Create and append the parameters to the command object
Set prEmpno = objCmd.CreateParameter("prEmpno", adSmallInt, adParamInput, ,8521)
' prEname is bound to a NVARCHAR column in the EMP table
Set prEname = objCmd.CreateParameter("prEname", adBSTR, adParamInput, , "Joe")
objCmd.Parameters.Append prEmpno
objCmd.Parameters.Append prEname

' Enabling the NDatatype property indicates to the provider
' that one or more of the bound parameters is of N datatype
objCmd.Properties("NDatatype") = TRUE

' Assume column ENAME in table EMP is of NVARCHAR type
```

```
objCmd.CommandText = "INSERT INTO EMP (EMPNO, ENAME) VALUES (?, ?)"

' Execute the SQL
objCmd.Execute

' It is a good idea to disable the property after execute as the same command
' object may be used for a different SQL statement
objCmd.Properties("NDatatype") = FALSE
```

SPPrmsLOB

This property allows the consumer to specify whether one or more of the parameters bound to the stored procedures are of Oracle's LOB datatype (`CLOB`, `BLOB`, or `NCLOB`). OraOLEDB requires this property to be set to `TRUE`, in order to fetch the parameter list of the stored procedure prior to execution. The use of this property limits the processing overhead to stored procedures having one or more LOB datatype parameters. This property should be set to `FALSE` after the command has been executed. By default, the property is set to `FALSE`.

Note: Consumers are required to use the ODBC procedure call escape sequence to call stored procedures or functions having LOB datatype parameters.

Example: Setting the Custom Property SPPrmsLOB

```
Dim objCon As NEW ADODB.Connection
Dim objCmd As NEW ADODB.Command
Dim prCLOB As NEW ADODB.Parameter
...
objCmd.ActiveConnection = objCon
objCmd.CommandType = adCmdText

' Create and append the parameters to the command object
Set prCLOB = objCmd.CreateParameter("prCLOB", adLongVarchar, adParamOutput, _
                                                10000)
objCmd.Parameters.Append prCLOB

' Enabling the SPPrmsLOB property indicates to the provider
' that one or more of the bound parameters is of LOB datatype
objCmd.Properties("SPPrmsLOB") = TRUE

' Assume the Stored Procedure requires a CLOB parameter
objCmd.CommandText = "{ call storedproc(?) }"
```

```
'Execute the SQL
objCmd.Execute

' It is a good idea to disable the property after execute as the
' same command object may be used for a different SQL statement
objCmd.Properties("SPPrmsLOB") = FALSE
```

Stored Procedures and Functions Returning Rowsets

Oracle Provider for OLE DB allows consumers to execute a **PL/SQL** stored procedure with an argument of REF CURSOR type or a stored function returning a REF CURSOR.

OraOLEDB returns a rowset for the REF CURSOR bind variable. Because there is no predefined datatype for REF CURSOR in the OLE DB specification, the consumer must not bind this parameter.

If the PL/SQL stored procedure has one or more arguments of REF CURSOR type, OraOLEDB binds these arguments appropriately and returns a rowset for each argument of REF CURSOR type.

If the PL/SQL stored function returns a REF CURSOR or has an argument of REF CURSOR type, OraOLEDB binds these appropriately and returns a rowset for each REF CURSOR type.

To use this feature, stored procedures or functions must be called in the ODBC procedure call escape sequence.

The stored procedure or function being called could be either standalone or packaged. However, the REF CURSOR being returned must be explicitly defined in a package in the database.

Multiple Rowsets

OraOLEDB supports returning more than one rowset from a stored procedure. Consumers can use this feature to access all the REF CURSORs being returned by a stored procedure.

Example: Stored Procedure Returning Multiple Rowsets

PL/SQL Package

```
CREATE OR REPLACE PACKAGE Employees AS
  TYPE empcur IS REF CURSOR;

  PROCEDURE GetEmpRecords(p_cursor OUT empcur,
                          q_cursor OUT empcur,
                          indeptno IN NUMBER,
                          p_errorcode OUT NUMBER);

  FUNCTION GetDept(inempno IN NUMBER,
                   p_errorcode OUT NUMBER)
    RETURN empcur;
END Employees;

CREATE OR REPLACE PACKAGE BODY Employees AS

  PROCEDURE GetEmpRecords(p_cursor OUT empcur,
                          q_cursor OUT empcur,
                          indeptno IN NUMBER,
                          p_errorcode OUT NUMBER) IS
  BEGIN
    p_errorcode := 0;
    OPEN p_cursor FOR
      SELECT *
      FROM emp
      WHERE deptno = indeptno
      ORDER BY empno;

OPEN q_cursor FOR
      SELECT empno
      FROM emp
      WHERE deptno = indeptno
      ORDER BY empno;

  EXCEPTION
    WHEN OTHERS THEN
      p_errorcode:= SQLCODE;

  END GetEmpRecords;

  FUNCTION GetDept(inempno IN NUMBER,
                   p_errorcode OUT NUMBER)
    RETURN empcur IS
      p_cursor empcur;
  BEGIN
    p_errorcode := 0;
    OPEN p_cursor FOR
      SELECT deptno
      FROM emp
      WHERE empno = inempno;
    RETURN (p_cursor);

  EXCEPTION
    WHEN OTHERS THEN
      p_errorcode:= SQLCODE;

  END GetDept;

END Employees;
```

ADO Program

```
Dim Con   As New ADODB.Connection
Dim Rst1  As New ADODB.Recordset
Dim Rst2  As New ADODB.Recordset
Dim Rst3  As New ADODB.Recordset
Dim Cmd   As New ADODB.Command
Dim Prm1  As New ADODB.Parameter
Dim Prm2  As New ADODB.Parameter

Con.Provider = "OraOLEDB.Oracle"
Con.ConnectionString = "Data Source=MyOraDb;" & _
                       "User ID=scott;Password=tiger;"
Con.Open
Cmd.ActiveConnection = Con

' Although Employees.GetEmpRecords() takes four parameters, only
' two need to be bound because Ref cursor parameters are automatically
' bound by the provider.

Set Prm1 = Cmd.CreateParameter("Prm1", adSmallInt, adParamInput, , 30)
Cmd.Parameters.Append Prm1
Set Prm2 = Cmd.CreateParameter("Prm2", adSmallInt, adParamOutput)
Cmd.Parameters.Append Prm2

' Enable PLSQLRSet property
Cmd.Properties ("PLSQLRSet") = TRUE
```

```
' Stored Procedures returning resultsets must be called using the
' ODBC escape sequence for calling stored procedures.
Cmd.CommandText = "{CALL Employees.GetEmpRecords(?, ?)}"

' Get the first recordset
Set Rst1 = Cmd.Execute

' Disable PLSQLRSet property
Cmd.Properties("PLSQLRSet") = FALSE

' Get the second recordset
Set Rst2 = Rst1.NextRecordset

' Just as in a stored procedure, the REF CURSOR return value must
' not be bound in a stored function.
Prm1.Value = 7839
Prm2.Value = 0

' Enable PLSQLRSet property
Cmd.Properties("PLSQLRSet") = TRUE

' Stored Functions returning resultsets must be called using the
' ODBC escape sequence for calling stored functions.
Cmd.CommandText = "{CALL Employees.GetDept(?, ?)}"

' Get the rowset
Set Rst3 = Cmd.Execute

' Disable PLSQLRSet
Cmd.Properties ("PLSQLRSet") = FALSE

' Clean up
Rst1.Close
Rst2.Close
Rst3.Close
```

Rowsets

To Create Rowsets

OraOLEDB supports `IOpenRowset::OpenRowset` and `ICommand::Execute` for creating rowsets.

To Create Rowsets with IOpenRowset::OpenRowset

When using `IOpenRowset::OpenRowset`, note the following guidelines:

- The `pTableID` parameter must contain a `DBID` structure that specifies a base table or a view.
- The `DBID` structure's `eKind` member must be set to `DBKIND_GUID_NAME`, `DBKIND_NAME`, or `DBKIND_PGUID_NAME`.
- The `DBID` structure's `uName` member must specify the base table or view name as a Unicode character string. It cannot be `NULL`.
- The `pIndexID` parameter of OpenRowset must be `NULL`.

To Create Rowsets with ICommand::Execute

OraOLEDB supports SQL `SELECT` statements that return rowsets. OraOLEDB also supports returning rowsets from PL/SQL stored procedures and functions.

By default, ADO creates a non-updatable rowset from a command object. An updatable rowset can be created by setting the `Updatability` and `IRowsetChange` properties on the command object. The `Updatability` property can be set to the following values:

Table 2–2 Possible Values for Updatability Property

Value	Description
1	update
2	delete
3	update and delete
4	insert
5	insert and update
6	insert and delete
7	insert, delete, and update

The following ADO code sample sets the `Updatability` property on a command object to allow insert, delete, and update operations on the rowset object.

```
Dim Cmd As New ADODB.Command
Dim Rst As New ADODB.Recordset
Dim Con As New ADODB.Connection
...
Cmd.ActiveConnection = Con
Cmd.CommandText = "SELECT * FROM emp"
Cmd.CommandType = adCmdText
cmd.Properties("IRowsetChange") = TRUE
Cmd.Properties("Updatability") = 7
’ creates an updatable rowset
Set Rst = cmd.Execute
```

Updatability

OraOLEDB supports both immediate and deferred update mode. However, insert and update operations cannot be deferred when the operation changes a non-scalar column, such as `LONG`, `BLOB`, or `CLOB`. When non-scalar column values are changed in a deferred update mode, the entire row is transmitted to the database as though the operation was in an immediate update mode. In addition, these operations cannot be undone with the `Undo` method (ADO) or `IRowsetUpdate::Undo()`. But if they are in a transaction, they can be rolled back with RollbackTrans method (ADO) or `ITransactionLocal::Abort()`.

Rowsets created using queries with joins are updatable by OraOLEDB only with the Client Cursor Engine enabled. C/C++ OLE DB consumers must enable this service to make these rowsets updatable. ADO consumers must specify the `CursorLocation` as `adUseClient` to make these rowsets updatable.

For example:

```
Dim objCon As New ADODB.Connection
Dim objRst As New ADODB.Recordset

objCon.Provider = "OraOLEDB.Oracle"
objCon.Open "MyOraDb", "scott", "tiger"
objRst.CursorLocation = adUseClient       ’ADO Client Cursor
objRst.Open "select ename, dname " & _
       "from emp, dept " & _
       "where emp.deptno = dept.deptno", _
       objCon, adOpenStatic, adLockOptimistic, adCmdText
```

```
'Recordset created is updatable. Please note that CursorLocation
'needs to be explicitly set to adUseClient for this join recordset
'to be updatable.
```

Server Data on Insert Property

If DBPROP_SERVERDATAONINSERT (Server Data on Insert) is set to TRUE using OraOLEDB, the consumer can obtain defaults, sequences, and triggered column values from newly inserted and updated rows, if the insert and update operations are made through the rowset.

Having DBPROP_SERVERDATAONINSERT set to TRUE may degrade performance for both insert and update executions using a rowset because OraOLEDB fetches row data from the database for the newly inserted and updated row. However, if DBPROP_SERVERDATAONINSERT is set to its default value of FALSE, only the explicitly provided values for insert and update operations get returned when column values are requested for those rows.

If the base table from which the rowset was created does not contain any defaults, sequences, or triggers, it is highly recommended that DBPROP_SERVERDATAONINSERT retain its default value of FALSE.

The DBPROP_SERVERDATAONINSERT property does not affect the performance of insert and update executions using the command object.

To Search for Rows with IRowsetFind::FindNext

OraOLEDB only supports searches performed on CHAR, DATE, FLOAT, NUMBER, RAW, and VARCHAR2 columns. Otherwise, DB_E_NOTSUPPORTED is returned.

When a search is done with a NULL value, only the DBCOMPAREOPS_EQ and DBCOMPAREOPS_NE compare operations are supported. Otherwise, DB_E_NOTSUPPORTED is returned.

OraOLEDB-specific Connection String Attributes for Rowsets

OraOLEDB-specific connection string attributes which affect the performance of the rowset are:

- CacheType - specifies the type of caching used by the provider to store rowset data. OraOLEDB provides two caching mechanisms:
  - Memory - The provider stores all the rowset data in-memory. This caching mechanism provides better performance at the expense of higher memory utilization. The default is Memory.

    - File - The provider stores all the rowset data on disk. This caching mechanism limits the memory consumption at the expense of performance.
- `ChunkSize` - This attribute specifies the size, in bytes, of the data in `LONG` and `LONG RAW` columns fetched and stored in the provider cache. Providing a high value for this attribute improves performance, but requires more memory to store the data in the rowset. Valid values are `1` to `65535`. The default is `100`.
- `FetchSize` - specifies the number of rows the provider will fetch at a time (fetch array). It must be set appropriately depending on the data size and the response time of the network. If the value is set too high, this could result in more wait time during the execution of the query. If the value is set too low, this could result in many more round trips to the database. Valid values are `1` to `429,496,296`. The default is `100`.

The default attributes values are set in the registry. For more information, see "Default Attribute Values" on page 2-4. The following ADO code example overrides the default attribute values:

```
Dim con As ADODB.Connection
Set con = NEW ADODB.Connection
con.ConnectionString = "Provider=OraOLEDB.Oracle;User ID=scott;" & _
                          "Password=tiger;Data Source=MyOraDB;" & _
                          "FetchSize=200;CacheType=File;"
con.Open
```

Tips for ADO Programmers

Setting the ADO Rowset property `LockType` to `adLockPessimistic` is not supported by Oracle Provider for OLE DB. If `LockType` is set to `adLockPessimistic`, OraOLEDB behaves similar to when set as `adLockOptimistic`. This behavior occurs because OraOLEDB does not perform explicit locks on the rows being modified. However, when the new data is submitted to the database, it only performs the update if the rowset data was not already updated by another user, which means that dirty writes are not allowed. LockType values `adLockReadOnly`, `adLockBatchOptimistic`, and `adLockOptimistic` are supported by OraOLEDB.

Setting ADO Rowset property `CursorType` to `adOpenKeyset` or `adOpenDynamic` is not supported by Oracle Provider for OLE DB. OraOLEDB does not support either of the two as Oracle supports *Statement Level Read Consistency*, which ensures that the data returned by a query contains only committed data as of the time the query was executed. `CursorType` values `adOpenStatic` and `adOpenForwardOnly` are supported by OraOLEDB.

Schema Rowsets

The schema rowsets available through Oracle Provider for OLE DB are:

- DBSCHEMA_COLUMNS
- DBSCHEMA_INDEXES
- DBSCHEMA_SCHEMATA
- DBSCHEMA_VIEWS
- DBSCHEMA_TABLES
- DBSCHEMA_PROVIDER_TYPES (forward scroll only)
- DBSCHEMA_FOREIGN_KEYS
- DBSCHEMA_PRIMARY_KEYS
- DBSCHEMA_PROCEDURES
- DBSCHEMA_PROCEDURE_PARAMETERS

Date Formats

The date format for the Oracle session cannot be set using ALTER SESSION SET NLS_DATE_FORMAT command. In Visual Basic, the date formats are controlled by the Regional Settings properties in the Windows Control Panel. For more information on Visual Basic date formats, refer to your Visual Basic documentation.

For Oracle Provider for OLE DB, NLS_DATE_FORMAT is fixed for the session to 'YYYY-MM-DD HH24:MI:SS' by the provider. If you pass the date to Oracle as a string, the date must be in the 'YYYY-MM-DD HH24:MI:SS' format. For example:

```
SELECT * FROM EMP WHERE HIREDATE > '1981-06-15 17:32:12'
```

To use a different format, you need to use the SQL function, TO_DATE(), to specify the format for dates passed as strings. For example:

```
SELECT * FROM EMP WHERE HIREDATE > TO_DATE('15-JUN-81', 'DD-MON-YY')
```

However, for dates passed as parameters, the date format is controlled by ADO, which is controlled by the Regional Settings in the Windows Control Panel. In this case, TO_DATE() should not be used. For example:

```
Private Sub Command1_Click()
  Dim objCon As New ADODB.Connection
  Dim objCmd As New ADODB.Command
  Dim objRst  As New ADODB.Recordset
  Dim pDate As New ADODB.Parameter

  objCon.Provider = "OraOLEDB.Oracle"
  objCon.Open "MyOraDb", "scott", "tiger"
  Set pDate = objCmd.CreateParameter("pDate", adDate, adParamInput)
  objCmd.Parameters.Append pDate
  objCmd.CommandText = _
         "SELECT * FROM EMP WHERE HIREDATE > ?"
  objCmd.ActiveConnection = objCon
  objCmd.CommandType = adCmdText
  pDate.Value = "06/15/1981"
  Set objRst = objCmd.Execute

    ...
End Sub
```

Case of Object Names

The names of all objects (tables, columns, views, and so forth) in Oracle are case-sensitive. This allows the two objects EMP and emp to exist in the same namespace in the database.

The query, SELECT ename FROM emp, executes correctly even though the table name is EMP (all uppercase) in the database. However, if you want to specify object names in mixed case, you can do so by enclosing the name in double quotes. For example:

```
SELECT ename FROM "Emp"
```

will execute successfully if the table name in the database is Emp. Double quotes preserve the case of the object names in Oracle.

LOB Support

The ISequentialStream interface is supported for all LONG, LONG RAW, and LOB (BLOB, CLOB, NCLOB, and BFILE) columns. The consumer can use this interface to read and write to all the LOB columns, except BFILE which is read-only. To have read and write access to these columns, the SELECT SQL statement used to create the rowset should not contain a join.

Note: Although most of the LOB columns in an Oracle database support up to 4 GB of data storage, ADO limits the maximum column size to 2 GB.

Columns having the `BFILE` datatype are not updatable in the `Rowset` interface. However, these columns can be updated using the Command interface, if the update is limited to modifying the directory and name of the external file pointed to by the `BFILE` column. For example:

```
INSERT INTO topomaps (areanum, topomap)
VALUES (158, BFILENAME('mapdir', 'topo158.tps'))
```

For more information on LOBs, see *Oracle9i Application Developer's Guide - Large Objects (LOBs)*.

Unicode Support

OraOLEDB supports the Unicode character set. Using this feature, consumers can use OraOLEDB to access data in multiple languages on the same client computer. It can be especially useful in creating global Internet applications supporting as many languages as the Unicode standard entails. For example, you can write a single Active Server Page (ASP) that accesses an Oracle9*i* database to dynamically generate contents in Japanese, Arabic, English, and Thai.

Types of Unicode Encoding

The Oracle databases store the Unicode data in the UTF8 encoding scheme, which is an ASCII compatible multibyte encoding of Unicode. Microsoft Windows 2000 and NT 4.0 use the UCS2 encoding, which is a 2-byte fixed width encoding scheme. OraOLEDB transparently converts the data between the two encoding schemes allowing the consumers to deal with only UCS2.

Note: The Unicode support is transparent to ADO consumers. OLE DB consumers using C/C++ need to explicitly specify `DBTYPE_WSTR` in their datatype bindings when Unicode data in involved.

How Oracle Unicode Support Works

OraOLEDB works in two modes, Unicode mode and non-Unicode mode. When the client character set is not a superset of the server character set, OraOLEDB automatically enables the Unicode mode. In this mode, OraOLEDB stores the data in its cache in the UCS2 encoding scheme. The user should ensure that the database's character set is UTF8 in order to prevent any data loss.

If the client character set is a superset of the server's, the provider operates in the non-Unicode mode. This mode provides slightly better performance as it does not have to deal with larger character buffers required by the UCS2 encoding.

The detection of the client's and the server's character set is performed during logon.

Note: OraOLEDB no longer requires the client character set to be set to UTF8 to enable the Unicode mode. The provider still supports such setups but no longer requires it.

See "Datatype Mappings in Rowsets and Parameters" on page A-2 for further information.

Unicode Support Setup

In order to prevent any data loss, the database character set should be UTF8. Other than this, there is no other setup required for the Unicode support.

Database Setup You must ensure that the Oracle database is configured to store the data in the UTF8 character set. The character set configuration is typically specified during database creation. To check the character set setting of your database, execute the following query in SQL*Plus:

```
SQL> SELECT parameter, value FROM nls_database_parameters
     WHERE parameter = 'NLS_CHARACTERSET';
```

If the character set of your database is not UTF8, you need to create a new database with the UTF8 character set and import your data into it. See *Oracle9i Database Administrator's Guide* for more information.

See Also:

Oracle9i Database Globalization Support Guide for general information

Errors

OLE and COM objects report errors through the `HRESULT` return code of the object member functions. An OLE/COM `HRESULT` is a bit-packed structure. OLE provides macros that dereference structure members. OraOLEDB exposes `IErrorLookup` to retrieve information about an error.

All objects support extended error information. For this, the consumer must instantiate the OLE DB Extended Error object followed by calling the method `GetErrorDescription()` to get the error text.

```
// Instantiate OraOLEDBErrorLookup and obtain a pointer to its
//   IErrorLookup interface
CoCreateInstance(CLSID_OraOLEDBErrorLookup, NULL, CLSCTX_INPROC_SERVER,
                 IID_IErrorLookup, (void **)&pIErrorLookup)
//Call the method GetErrorDescription() to get the full error text
pIErrorLookup->GetErrorDescription()
```

The OraOLEDB provider returns the entire error stack in one text block.

For ADO users, the following example applies:

```
Dim oerr As ADODB.Error
For Each oerr in con.Errors
    MsgBox "Error: " & oerr.Description & vbCrLf _
        & "Source: " & oerr.Source
Next
```

OLE DB .NET Data Provider Compatibility

The OLE DB .NET Data Provider can utilize OraOLEDB as the OLE DB Provider for accessing Oracle.

To make OraOLEDB compatible with OLE DB .NET Data Provider, set the connection string attribute `OLEDB.NET` to `True`.

Setting the `OLEDB.NET` attribute to `False` disables .NET compatibility.

Note: The `OLEDB.NET` connection string attribute must not be used in ADO applications.

Using the OLEDB.NET Attribute in a Connection String

When using OraOLEDB with the OLE DB .NET Data Provider, the `OLEDB.NET` connection attribute must be set to `True` as shown in the following examples:

```
// in VB.NET
Dim con As New OleDbConnection()
con.ConnectionString = "Provider=OraOLEDB.Oracle;User Id=scott;" & _
     "Password=tiger;Data Source=Oracle;OLEDB.NET=True;"
```

```
con.Open

// in C#
...
OleDbConnection con = new OleDbConnection();
con.ConnectionString = "Provider=OraOLEDB.Oracle;User Id=scott;" +
     "Password=tiger;Data Source=Oracle;OLEDB.NET=true;"
con.Open();
...
```

Using OraOLEDB Custom Properties

ADO allows OraOLEDB provider-specific properties to be set at the object level. The OraOLEDB-specific properties `SPPrmsLOB` and `NDatatype` can only be set at the ADO command object level, as shown in the following example:

```
// in VB
Dim cmd as new ADODB.Command
...
cmd.Properties("SPPrmsLOB") = True
cmd.Properties("NDatatype") = True
...
```

However, the OLE DB .NET Data Provider cannot expose OLE DB provider-specific properties at the object level. Therefore, the `SPPrmsLOB` and `NDatatype` properties can only be set as connection string attributes when OraOLEDB is used by OLE DB .NET Data Provider:

```
// in VB.NET
Dim con As New OleDbConnection()
con.ConnectionString = "Provider=OraOLEDB.Oracle;User Id=scott;" & _
     "Password=tiger;Data Source=Oracle;OLEDB.NET=True;" & _
     "SPPrmsLOB=False;NDatatype=False;"
con.Open()
```

Both `SPPrmsLOB` and `NDatatype` connection string attributes are set to `False` by default if they are not specified.

Setting either of these connection string attributes to `True` incurs additional processing overhead when executing commands with parameters. For this reason, before setting either attribute to True, see "OraOLEDB Custom Properties for Commands" on page 2-8.

Updating Oracle with DataTable Changes

In order for the OleDbDataAdapter.Update() method to properly update Oracle with changes made in the DataTable, the DataTable must contain a primary key of a database table. If the database table does not contain a primary key, the ROWID must be selected explicitly when populating the DataTable, so that the ROWID can be used to uniquely identify a row when updating a row in the database.

Do not select the ROWID from database tables that contains a primary key. If the ROWID is selected along with a primary key, the ROWID will be the only column marked as the primary key.

See Also: For further information on using the OLE DB .NET Data Provider

- Microsoft .NET Documentation
- Microsoft .NET Framework Class Library

Using OraOLEDB with Visual Basic

The following simple example illustrates how to use Oracle Provider for OLE DB with ADO in Visual Basic 6.0 to connect to an Oracle database and execute PL/SQL stored procedures and functions.

Setting Up the Oracle Database

This example assumes that the Oracle database has the demonstration table `EMP` under the user account `scott`. The `scott` account is included in the Oracle starter database. If the account does not exist on your database, create the account before running the sample program. If your database does not contain the `emp` table, you can use the `demobld.sql` script to create the demonstration tables.

This example also uses *`exampledb`* as the database network alias when connecting to the Oracle database. You will need to change this network alias to match your system.

Step 1 Build the sample tables:

1. Start SQL*Plus.
2. Connect as username `scott` with the password `tiger`.
3. Run the `demobld.sql` script:

```
SQL> @ORACLE_BASE\ORACLE_HOME\sqlplus\demo\demobld.sql;
```

After the `emp` table has been created in the `scott` account, you need to create the PL/SQL package that contains the stored procedure and function that are run in the Visual Basic example.

Step 2 Create the PL/SQL package:

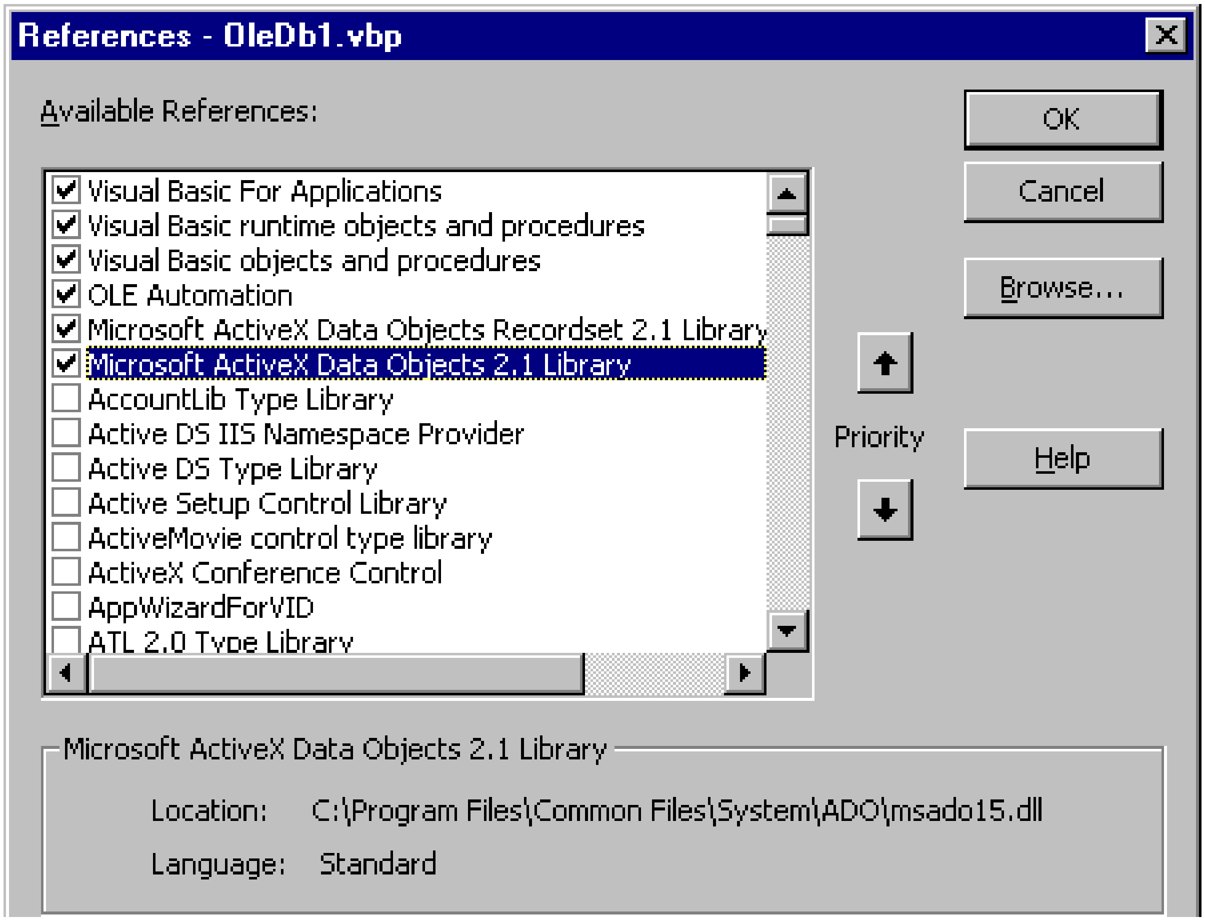
1. Start SQL*Plus.
2. Connect as username `scott` with the password `tiger`.
3. Create the PL/SQL packages shown in "PL/SQL Package" on page 2-13.

Note: When creating PL/SQL packages the `/` character is used as a terminator and must be added on a separate line following each CREATE PACKAGE...END block.

Setting Up the Visual Basic Project

After the Oracle database setups are completed, you can create the Visual Basic 6.0 project.

1. Start Visual Basic 6.0 and create a new project.
2. Make sure that the Microsoft ActiveX Data Objects 2.1 Library and Microsoft ActiveX Data Objects Recordset 2.1 Library are included as Project References.



Available References:

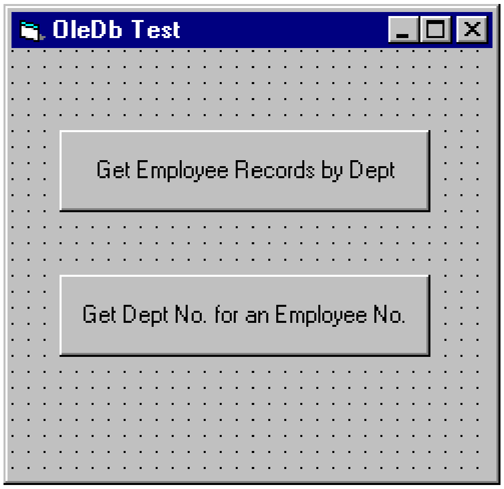
- ✓ Visual Basic For Applications
- ✓ Visual Basic runtime objects and procedures
- ✓ Visual Basic objects and procedures
- ✓ OLE Automation
- ✓ Microsoft ActiveX Data Objects Recordset 2.1 Library
- ✓ Microsoft ActiveX Data Objects 2.1 Library
- ☐ AccountLib Type Library
- ☐ Active DS IIS Namespace Provider
- ☐ Active DS Type Library
- ☐ Active Setup Control Library
- ☐ ActiveMovie control type library
- ☐ ActiveX Conference Control
- ☐ AppWizardForVID
- ☐ ATL 2.0 Type Library

Microsoft ActiveX Data Objects 2.1 Library

Location: C:\Program Files\Common Files\System\ADO\msado15.dll

Language: Standard

3. Add two commands buttons to the form. One of the buttons will run the code to execute the PL/SQL procedure `GetEmpRecords`. The other will run the code to execute the PL/SQL function `GetDept`.


OleDb Test
Get Employee Records by Dept
Get Dept No. for an Employee No.

4. Add the following code to Click subroutine of the button that will run the code to execute the PL/SQL procedure `GetEmpRecords`.

```
Dim Oracon As ADODB.Connection
Dim recset As New ADODB.Recordset
Dim cmd As New ADODB.Command
Dim param1 As New ADODB.Parameter
Dim param2 As New ADODB.Parameter
Dim objErr As ADODB.Error
Dim Message, Title, Default, EmpNoValue

Message = "Enter an employee number (5000 - 9000)"
Title = "Choose an Employee"
Default = "7654"

On Error GoTo err_test

EmpNoValue = InputBox(Message, Title, Default)
If EmpNoValue = "" Then Exit Sub
If EmpNoValue < 5000 Or EmpNoValue > 9000 Then EmpNoValue = 7654

Set Oracon = CreateObject("ADODB.Connection")
Oracon.ConnectionString = "Provider=OraOLEDB.Oracle;" & _
                          "Data Source=exampledb;" & _
                          "User ID=scott;" & _
                          "Password=tiger;"

Oracon.Open
```

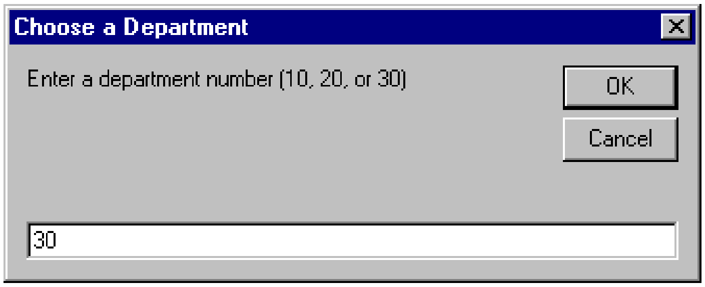
```
Set cmd.ActiveConnection = Oracon
Set param1 = cmd.CreateParameter("param1", adSmallInt, adParamInput, ,
                                 EmpNoValue)
cmd.Parameters.Append param1
Set param2 = cmd.CreateParameter("param2", adSmallInt, adParamOutput)
cmd.Parameters.Append param2

’ Enable PLSQLRSet property
Cmd.Properties ("PLSQLRSet") = TRUE

cmd.CommandText = "{CALL Employees.GetDept(?, ?)}"
Set recset = cmd.Execute

’ Disable PLSQLRSet property
Cmd.Properties ("PLSQLRSet") = FALSE

MsgBox "Number: " & EmpNoValue & "  Dept: " & recset.Fields("deptno").Value

Exit Sub

err_test:
    MsgBox Error$
    For Each objErr In Oracon.Errors
        MsgBox objErr.Description
    Next
    Oracon.Errors.Clear
    Resume Next
```

5. Add the following code to Click subroutine of the button that will run the code to execute the PL/SQL function `GetDept`.

```
Dim Oracon As ADODB.Connection
Dim recset As New ADODB.Recordset
Dim cmd As New ADODB.Command
Dim param1 As New ADODB.Parameter
Dim param2 As New ADODB.Parameter
Dim objErr As ADODB.Error

Dim Message, Title, Default, DeptValue
Message = "Enter a department number (10, 20, or 30)"
Title = "Choose a Department"
Default = "30"

On Error GoTo err_test
DeptValue = InputBox(Message, Title, Default)
```

```
If DeptValue = "" Then Exit Sub
If DeptValue < 10 Or DeptValue > 30 Then DeptValue = 30

Set Oracon = CreateObject("ADODB.Connection")
Oracon.ConnectionString = "Provider=OraOLEDB.Oracle;" & _
                          "Data Source=exampledb;" & _
                          "User ID=scott;" & _
                          "Password=tiger;"
Oracon.Open
Set cmd = New ADODB.Command
Set cmd.ActiveConnection = Oracon
Set param1 = cmd.CreateParameter("param1", adSmallInt, adParamInput, ,
                                DeptValue)
cmd.Parameters.Append param1
Set param2 = cmd.CreateParameter("param2", adSmallInt, adParamOutput)
cmd.Parameters.Append param2

’ Enable PLSQLRSet property
Cmd.Properties ("PLSQLRSet") = TRUE

cmd.CommandText = "{CALL Employees.GetEmpRecords(?, ?)}"
Set recset = cmd.Execute

’ Disable PLSQLRSet property
Cmd.Properties ("PLSQLRSet") = FALSE

Do While Not recset.EOF
   MsgBox "Number: " & recset.Fields("empno").Value & "  Name: " &
   recset.Fields("ename").Value & "  Dept: " & recset.Fields("deptno").Value
   recset.MoveNext
Loop

Exit Sub

err_test:
    MsgBox Error$
    For Each objErr In Oracon.Errors
        MsgBox objErr.Description
    Next
    Oracon.Errors.Clear
    Resume Next
```

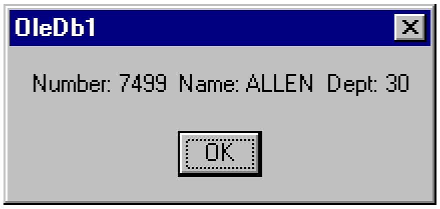
6. Run the project and check the results. For example, if you choose the Get Employee Records by Dept button, you would get a dialog box requesting that you enter a department number.



Enter a department number (10, 20, or 30)

30

Once you have entered a department number and OK, another dialog box displays employee names and numbers from that department.



Number: 7499 Name: ALLEN Dept: 30

A
Provider-Specific Information

This appendix describes OLE DB information that is specific to Oracle Provider for OLE DB. For generic OLE DB information that includes a detailed listing of all OLE DB properties and interfaces, see the Microsoft *OLE DB Programmer's Reference Guide*.

This appendix contains these topics:

- Datatype Mappings in Rowsets and Parameters
- Properties Supported
- Interfaces Supported
- MetaData Columns Supported
- OraOLEDB Tracing

Datatype Mappings in Rowsets and Parameters

This section lists the datatype mappings between Oracle datatypes and OLE DB-defined types. Oracle Provider for OLE DB represents Oracle datatypes by using certain OLE DB-defined datatypes in the rowset as well as in parameters. OLE DB-defined types are also mapped to an Oracle datatype when creating tables.

Each Oracle datatype is mapped to a specific OLE DB datatype, as shown in Table A–1. This correspondence is used when datatype information is retrieved from an Oracle database.

Table A–1 Datatype Mappings

Oracle Datatype	OLE DB Datatype - Regular (Non Unicode) Mode	OLE DB Datatype - Unicode Mode
BFILE	DBTYPE_BYTES	DBTYPE_BYTES
BLOB	DBTYPE_BYTES	DBTYPE_BYTES
CHAR	DBTYPE_STR	DBTYPE_WSTR
CLOB	DBTYPE_STR	DBTYPE_WSTR
DATE	DBTYPE_DBTIMESTAMP	DBTYPE_DBTIMESTAMP
FLOAT	DBTYPE_R8	DBTYPE_R8
LONG	DBTYPE_STR	DBTYPE_WSTR
LONG RAW	BTYPE_BYTES	DBTYPE_BYTES
NCHAR	DBTYPE_STR	DBTYPE_WSTR
NCLOB	DBTYPE_STR	DBTYPE_WSTR
NUMBER	DBTYPE_VARNUMERIC	DBTYPE_VARNUMERIC
NUMBER(p,s)	DBTYPE_NUMERIC	DBTYPE_NUMERIC
NVARCHAR2	DBTYPE_STR	DBTYPE_WSTR
RAW	DBTYPE_BYTES	DBTYPE_BYTES
ROWID	DBTYPE_STR	DBTYPE_STR
VARCHAR	DBTYPE_STR	DBTYPE_WSTR

Properties Supported

This section lists the properties supported by Oracle Provider for OLE DB. The read / write status and initial values are noted.

- Data Source Properties
- Data Source Info Properties
- Initialization and Authorization Properties
- Session Properties
- Rowset Properties

Data Source Properties

Table A–2 DBPROPSET_DATASOURCE Properties

Property	Status	Initial Value
DBPROP_CURRENTCATALOG	READ-ONLY	NULL

Data Source Info Properties

Table A–3 DBPROPSET_DATASOURCEINFO Properties

Property	Status	Initial Value
DBPROP_ACTIVESESSIONS	READ-ONLY	0, Unlimited sessions
DBPROP_ASYNCTXNABORT	READ-ONLY	VARIANT_FALSE
DBPROP_ASYNCTXNCOMMIT	READ-ONLY	VARIANT_FALSE
DBPROP_BYREFACCESSORS	READ-ONLY	VARIANT_TRUE
DBPROP_CATALOGLOCATION	READ-ONLY	DBPROPVAL_CL_END
DBPROP_CATALOGTERM	READ-ONLY	"Database link"
DBPROP_CATALOGUSAGE	READ-ONLY	DBPROPVAL_CU_DML_STATEMENTS
DBPROP_COLUMNDEFINITION	READ-ONLY	DBPROPVAL_CD_NOTNULL
DBPROP_CONCATNULLBEHAVIOR	READ-ONLY	DBPROPVAL_CB_NON_NULL
DBPROP_CONNECTIONSTATUS	READ-ONLY	DBPROPVAL_CS_INITIALIZED
DBPROP_DATASOURCENAME	READ-ONLY	" ", set at runtime
DBPROP_DATASOURCEREADONLY	READ-ONLY	VARIANT_FALSE

Table A–3 DBPROPSET_DATASOURCEINFO Properties

Property	Status	Initial Value
DBPROP_DBMSNAME	READ-ONLY	" ", set at runtime
DBPROP_DBMSVER	READ-ONLY	set at runtime
DBPROP_DSOTHREADMODEL	READ / WRITE	DBPROPVAL_RT_FREETHREAD
DBPROP_GROUPBY	READ-ONLY	DBPROPVAL_GB_CONTAINS_SELECT
DBPROP_HETEROGENEOUSTABLES	READ-ONLY	DBPROPVAL_HT_DIFFERENT_CATALOGS
DBPROP_IDENTIFIERCASE	READ-ONLY	DBPROPVAL_IC_UPPER
DBPROP_MAXINDEXSIZE	READ-ONLY	0, limit unknown - depends on blocksize
DBPROP_MAXOPENCHAPTERS	READ-ONLY	0, not supported
DBPROP_MAXORSINFILTER	READ-ONLY	0, not supported
DBPROP_MAXROWSIZE	READ-ONLY	0, No limit
DBPROP_MAXROWSIZEINCLUDESBLOB	READ-ONLY	VARIANT_FALSE
DBPROP_MAXSORTCOLUMNS	READ-ONLY	0, not supported
DBPROP_MAXTABLESINSELECT	READ-ONLY	0, no limit
DBPROP_MULTIPLEPARAMSETS	READ-ONLY	VARIANT_TRUE
DBPROP_MULTIPLERESULTS	READ-ONLY	DBPROP_MR_SUPPORTED | DBPROPVAL__MR_CONCURRENT
DBPROP_MULTIPLESTORAGEOBJECTS	READ-ONLY	VARIANT_FALSE
DBPROP_MULTITABLEUPDATE	READ-ONLY	VARIANT_FALSE
DBPROP_NULLCOLLATION	READ-ONLY	DBPROPVAL_NC_HIGH
DBPROP_OLEOBJECTS	READ-ONLY	DBPROPVAL_OO_BLOB
DBPROP_ORDERBYCOLUMNSINSELECT	READ-ONLY	VARIANT_FALSE
DBPROP_OUTPUTPARAMETERAVAILABILITY	READ-ONLY	DBPROPVAL_OA_ATEXECUTE
DBPROP_PERSISTENTIDTYPE	READ-ONLY	DBPROPVAL_PT_NAME
DBPROP_PREPAREABORTBEHAVIOR	READ-ONLY	DBPROPVAL_CB_PRESERVE
DBPROP_PREPARECOMMITBEHAVIOR	READ-ONLY	DBPROPVAL_CB_PRESERVE
DBPROP_PROCEDURETERM	READ-ONLY	"PL / SQL Stored Procedure"
DBPROP_PROVIDERFRIENDLYNAME	READ-ONLY	"Oracle Provider for OLE DB"
DBPROP_PROVIDERNAME	READ-ONLY	OraOLEDB.dll
DBPROP_PROVIDEROLEDBVER	READ-ONLY	"02.01"

Table A–3 DBPROPSET_DATASOURCEINFO Properties

Property	Status	Initial Value
DBPROP_PROVIDERVER	READ-ONLY	set to current OraOLEDB version
DBPROP_QUOTEDIDENTIFIERCASE	READ-ONLY	DBPROPVAL_IC_SENSITIVE
DBPROP_ROWSETCONVERSIONSONCOMMAND	READ-ONLY	VARIANT_TRUE
DBPROP_SCHEMATERM	READ-ONLY	"Owner"
DBPROP_SCHEMAUSAGE	READ-ONLY	DBPROPVAL_SU_DML_STATEMENTS | DBPROPVAL_SU_TABLE_DEFINITION | DBPROPVAL_SU_INDEX_DEFINITION | DBPROPVAL_SU_PRIVILEGE_DEFINITION
DBPROP_SERVERNAME	READ-ONLY	" ", set at runtime
DBPROP_SORTONINDEX	READ-ONLY	VARIANT_FALSE
DBPROP_SQLSUPPORT	READ-ONLY	DBPROPVAL_SQL_ODBC_MINIMUM | DBPROPVAL_SQL_ANSI92_ENTRY | DBPROPVAL_SQL_ESCAPECLAUSES
DBPROP_STRUCTUREDSTORAGE	READ-ONLY	DBPROPVAL_SS_ISEQUENTIAL_STREAM
DBPROP_SUBQUERIES	READ-ONLY	DBPROPVAL_SQ_CORRELATEDSUBQUERIES
DBPROP_SUPPORTEDTXNDDL	READ-ONLY	DBPROPVAL_TC_DDL_COMMIT
DBPROP_SUPPORTEDTXNISOLEVELS	READ-ONLY	DBPROPVAL_TI_CURSORSTABILITY | DBPROPVAL_TI_READCOMMITTED
DBPROP_SUPPORTEDTXNISORETAIN	READ-ONLY	DBPROPVAL_TR_DONTCARE
DBPROP_TABLETERM	READ-ONLY	"Table"
DBPROP_USERNAME	READ-ONLY	" ", set at runtime

Initialization and Authorization Properties

Table A–4 DBPROPSET_DBINIT Properties

Property	Status	Initial Value
DBPROP_AUTH_PERSIST_SENSITIVE_AUTHINFO	READ-ONLY	VARIANT_FALSE
DBPROP_AUTH_USERID	READ / WRITE	User ID
DBPROP_INIT_DATASOURCE	READ / WRITE	Connect string
DBPROP_INIT_HWND	READ / WRITE	Window handle for prompt

Table A–4 DBPROPSET_DBINIT Properties

Property	Status	Initial Value
DBPROP_INIT_LCID	READ / WRITE	LCID of system
DBPROP_INIT_OLEDBSERVICES	READ / WRITE	DBPROPVAL_OS_ENABLEALL
DBPROP_INIT_PROMPT	READ / WRITE	DBPROMPT_NOPROMPT

Session Properties

Table A–5 DBPROPSET_SESSION Properties

Property	Status	Initial Value
DBPROP_SESS_AUTOCOMMITISOLEVELS	READ-ONLY	DBPROPVAL_TI_CURSORSTABILITY | DBPROPVAL_TI_READCOMMITTED

Rowset Properties

Table A–6 DBPROPSET_ROWSET Properties

Property	Status	Initial Value
DBPROP_ABORTPRESERVE	READ / WRITE	VARIANT_TRUE
DBPROP_ACCESSORORDER	READ-ONLY	DBPROP_AO_RANDOM
DBPROP_APPENDONLY	READ-ONLY	VARIANT_FALSE
DBPROP_BLOCKINGSTORAGEOBJECTS	READ-ONLY	VARIANT_FALSE
DBPROP_BOOKMARKINFO	READ-ONLY	0
DBPROP_BOOKMARKS	READ / WRITE	VARIANT_FALSE
DBPROP_BOOKMARKSKIPPED	READ / WRITE	VARIANT_TRUE
DBPROP_BOOKMARKTYPE	READ-ONLY	DBPROP_BMK_NUMERIC
DBPROP_CACHEDEFERRED	READ-ONLY	VARIANT_FALSE
DBPROP_CANFETCHBACKWARDS	READ / WRITE	VARIANT_FALSE
DBPROP_CANHOLDROWS	READ / WRITE	VARIANT_FALSE
DBPROP_CANSCROLLBACKWARDS	READ / WRITE	VARIANT_FALSE
DBPROP_CHANGEINSERTEDROWS	READ-ONLY	VARIANT_TRUE
DBPROP_CLIENTCURSOR	READ / WRITE	VARIANT_TRUE
DBPROP_COLUMNRESTRICT	READ-ONLY	VARIANT_TRUE

Table A–6 DBPROPSET_ROWSET Properties

Property	Status	Initial Value
DBPROP_COMMANDTIMEOUT	READ / WRITE	0, currently not operational
DBPROP_COMMITPRESERVE	READ / WRITE	VARIANT_TRUE
DBPROP_DEFERRED	READ-ONLY	VARIANT_TRUE
DBPROP_DELAYSTORAGEOBJECTS	READ-ONLY	VARIANT_TRUE, no delayed update
DBPROP_FINDCOMPAREOPS	READ-ONLY	DBPROPVAL_CO_EQUALITY | DBPROPVAL_CO_STRING | DBPROPVAL_CO_CASESENSITIVE | DBPROPVAL_CO_CASEINSENSITIVE | DBPROPVAL_CO_CONTAINS | DBPROPVAL_CO_BEGINSWITH
DBPROP_HIDDENCOLUMNS	READ-ONLY	0
DBPROP_IACCESSOR	READ-ONLY	VARIANT_TRUE
DBPROP_ICOLUMNSINFO	READ-ONLY	VARIANT_TRUE
DBPROP_ICOLUMNSROWSET	READ / WRITE	VARIANT_TRUE
DBPROP_ICONNECTIONPOINTCONTAINER	READ-ONLY	VARIANT_TRUE
DBPROP_ICONVERTTYPE	READ-ONLY	VARIANT_TRUE
DBPROP_IMMOBILEROWS	READ-ONLY	VARIANT_TRUE
DBPROP_IMULTIPLERESULTS	READ / WRITE	VARIANT_TRUE
DBPROP_IROWSET	READ-ONLY	VARIANT_TRUE
DBPROP_IROWSETCHANGE	READ / WRITE	VARIANT_FALSE
DBPROP_IROWSETFIND	READ / WRITE	VARIANT_FALSE
DBPROP_IROWSETIDENTITY	READ-ONLY	VARIANT_TRUE
DBPROP_IROWSETINFO	READ-ONLY	VARIANT_TRUE
DBPROP_IROWSETLOCATE	READ / WRITE	VARIANT_FALSE
DBPROP_IROWSETREFRESH	READ / WRITE	VARIANT_FALSE
DBPROP_IROWSETSCROLL	READ / WRITE	VARIANT_FALSE
DBPROP_IROWSETUPDATE	READ / WRITE	VARIANT_FALSE
DBPROP_ISEQUENTIALSTREAM	READ / WRITE	VARIANT_TRUE
DBPROP_ISUPPORTERRORINFO	READ / WRITE	VARIANT_TRUE
DBPROP_LITERALBOOKMARKS	READ-ONLY	VARIANT_FALSE
DBPROP_LITERALIDENTITY	READ-ONLY	VARIANT_FALSE

Table A–6 DBPROPSET_ROWSET Properties

Property	Status	Initial Value
DBPROP_LOCKMODE	READ-ONLY	DBPROPVAL_LM_NONE
DBPROP_MAXOPENROWS	READ / WRITE	0, No limit
DBPROP_MAXPENDINGROWS	READ-ONLY	0, No limit
DBPROP_MAXROWS	READ / WRITE	0
DBPROP_MAXROWSIZE	READ-ONLY	0
DBPROP_MAXROWSIZEINCLUDESBLOB	READ-ONLY	VARIANT_FALSE
DBPROP_NOTIFICATIONGRANULARITY	READ / WRITE	DBPROPVAL_NT_MULTIPLEROWS
DBPROP_NOTIFICATIONPHASES	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER | DBPROPVAL_NP_FAILEDTODO | DBPROPVAL_NP_DIDEVENT
DBPROP_NOTIFYCOLUMNSET	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER | DBPROPVAL_NP_FAILEDTODO | DBPROPVAL_NP_DIDEVENT
DBPROP_NOTIFYROWDELETE	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER | DBPROPVAL_NP_FAILEDTODO | DBPROPVAL_NP_DIDEVENT
DBPROP_NOTIFYROWFIRSTCHANGE	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO
DBPROP_NOTIFYROWINSERT	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER | DBPROPVAL_NP_FAILEDTODO | DBPROPVAL_NP_DIDEVENT
DBPROP_NOTIFYROWRESYNCH	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER
DBPROP_NOTIFYROWSETRELEASE	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER
DBPROP_NOTIFYROWSETFETCHPOSITIONCHANGE	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER

Table A–6 DBPROPSET_ROWSET Properties

Property	Status	Initial Value
DBPROP_NOTIFYROWUNDOCHANGE	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER | DBPROPVAL_NP_FAILEDTODO | DBPROPVAL_NP_DIDEVENT
DBPROP_NOTIFYROWUNDODELETE	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER | DBPROPVAL_NP_FAILEDTODO | DBPROPVAL_NP_DIDEVENT
DBPROP_NOTIFYROWUNDOINSERT	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER | DBPROPVAL_NP_FAILEDTODO | DBPROPVAL_NP_DIDEVENT
DBPROP_NOTIFYROWUNDOUPDATE	READ / WRITE	DBPROPVAL_NP_OKTODO | DBPROPVAL_NP_ABOUTTODO | DBPROPVAL_NP_SYNCHAFTER | DBPROPVAL_NP_FAILEDTODO | DBPROPVAL_NP_DIDEVENT
DBPROP_ORDEREDBOOKMARKS	READ-ONLY	VARIANT_TRUE
DBPROP_OTHERINSERT	READ-ONLY	VARIANT_FALSE
DBPROP_OTHERUPDATEDELETE	READ-ONLY	VARIANT_FALSE
DBPROP_OWNINSERT	READ-ONLY	VARIANT_TRUE
DBPROP_OWNUPDATEDELETE	READ-ONLY	VARIANT_TRUE
DBPROP_QUICKRESTART	READ / WRITE	VARIANT_FALSE
DBPROP_REENTRANTEVENTS	READ-ONLY	VARIANT_FALSE
DBPROP_REMOVEDELETED	READ-ONLY	VARIANT_TRUE
DBPROP_REPORTMULTIPLECHANGES	READ-ONLY	VARIANT_FALSE
DBPROP_RETURNPENDINGINSERTS	READ / WRITE	VARIANT_TRUE
DBPROP_ROWRESTRICT	READ / WRITE	VARIANT_FALSE
DBPROP_ROWTHREADMODEL	READ-ONLY	DBPROPVAL_RT_FREETHREAD
DBPROP_SERVERCURSOR	READ / WRITE	VARIANT_FALSE
DBPROP_SERVERDATAONINSERT	READ / WRITE	VARIANT_TRUE
DBPROP_STRONGIDENTITY	READ / WRITE	VARIANT_TRUE

Table A–6 DBPROPSET_ROWSET Properties

Property	Status	Initial Value
DBPROP_TRANSACTEDOBJECT	READ-ONLY	VARIANT_TRUE
DBPROP_UNIQUEROWS	READ/WRITE	VARIANT_FALSE
DBPROP_UPDATABILITY	READ/WRITE	DBPROPVAL_UP_CHANGE | DBPROPVAL_UP_DELETE | DBPROPVAL_UP_INSET

Rowset Property Implications

Oracle Provider for OLE DB sets other necessary properties if a particular property is set to VARIANT_TRUE.

- If DBPROP_IROWSETLOCATE is set to VARIANT_TRUE, the following properties are also set to VARIANT_TRUE:
    - DBPROP_IROWSETIDENTITY
    - DBPROP_CANHOLDROWS
    - DBPROP_BOOKMARKS
    - DBPROP_CANFETCHBACKWARDS
    - DBPROP_CANSCROLLBACKWARDS
- If DBPROP_IROWSETSCROLL is set to VARIANT_TRUE, the following properties are also set to VARIANT_TRUE:
    - DBPROP_IROWSETIDENTITY
    - DBPROP_IROWSETLOCATE
    - DBPROP_CANHOLDROWS
    - DBPROP_BOOKMARKS
    - DBPROP_CANFETCHBACKWARDS
    - DBPROP_CANSCROLLBACKWARDS
- If DBPROP_IROWSETUPDATE is set to VARIANT_TRUE, the following properties are also set to VARIANT_TRUE:
    - DBPROP_IROWSETCHANGE

Interfaces Supported

This section identifies the OLE DB interfaces that are supported by Oracle Provider for OLE DB.

- Data Source
- Session
- Command
- Rowset
- Multiple Results
- Transaction Options
- Custom Error Object

Data Source

```
CoType TDataSource {
   interface IDBCreateSession;
   interface IDBInitialize;
   interface IDBProperties;
   interface IPersist;
   interface IDBInfo;
   interface ISupportErrorInfo;
}
```

Session

```
CoType TSession {
   interface IGetDataSource;
   interface IOpenRowset;
   interface ISessionProperties;
   interface IDBCreateCommand;
   interface IDBSchemaRowset;
   interface ISupportErrorInfo;
   interface ITransactionJoin;
   interface ITransactionLocal;
   interface ITransaction;
}
```

Command

```
CoType TCommand {
   interface IAccessor;
   interface IColumnsInfo;
   interface ICommand;
   interface ICommandProperties;
   interface ICommandText;
   interface IConvertType;
   interface IColumnsRowset;
   interface ICommandPrepare;
   interface ICommandWithParameters;
   interface ISupportErrorInfo;
}
```

Rowset

```
CoType TRowset {
   interface IAccessor;
   interface IColumnsInfo;
   interface IConvertType;
   interface IRowset;
   interface IRowsetInfo;
   interface IColumnsRowset;
   interface IConnectionPointContainer;
   interface IRowsetChange;
   interface IRowsetFind;
   interface IRowsetIdentity;
   interface IRowsetLocate;
   interface IRowsetRefresh;
   interface IRowsetScroll;
   interface IRowsetUpdate;
   interface ISupportErrorInfo;
}
```

Multiple Results

```
CoType TMultipleResults {
   interface IMultipleResults;
   interface ISupportErrorInfo;
}
```

Transaction Options

```
CoType TTransactionOptions {
   interface ITransactionOptions;
   interface ISupportErrorInfo;
}
```

Custom Error Object

```
CoType TCustomErrorObject {
   interface IErrorLookup;
}
```

MetaData Columns Supported

The following metadata columns are supported by OraOLEDB's column rowset:

- DBCOLUMN_IDNAME
- DBCOLUMN_GUID
- DBCOLUMN_PROPID
- DBCOLUMN_NAME
- DBCOLUMN_NUMBER
- DBCOLUMN_TYPE
- DBCOLUMN_TYPEINFO
- DBCOLUMN_COLUMNSIZE
- DBCOLUMN_PRECISION
- DBCOLUMN_SCALE
- DBCOLUMN_FLAGS
- DBCOLUMN_BASECATALOGNAME
- DBCOLUMN_BASECOLUMNNAME
- DBCOLUMN_BASESCHEMANAME
- DBCOLUMN_BASETABLENAME
- DBCOLUMN_COMPUTEMODE

- `DBCOLUMN_ISAUTOINCREMENT`
- `DBCOLUMN_ISCASESENSITIVE`
- `DBCOLUMN_ISSEARCHABLE`
- `DBCOLUMN_OCTETLENGTH`
- `DBCOLUMN_KEYCOLUMN`

OraOLEDB Tracing

OraOLEDB provides the ability to trace the interface calls for debugging purposes. This feature has been provided to assist Oracle Support Services in debugging customer issues.

The provider can be configured to record the following information:

- For OLE DB Interface method entry and exit:
  - Parameter value(s) supplied (entry)
  - Return value; `HRESULT` (exit)
  - Thread ID (entry and exit)
- For Distributed transaction enlistment and delistment:
  - Session object information
  - Transaction ID

Note: In order to record global transaction enlistment and delistment information, the `TraceLevel` value must be set to session object. See "TraceLevel" on page A-15.

Registry Setting for Tracing Calls

In order to trace the interface calls, you must configure the following registry values for `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\OLEDB\`:

- `TraceFileName`

  Valid Value: Any valid path and filename

  `TraceFileName` specifies the filename that is to be used for logging trace information. If `TraceOption` is set to `0`, the name is used as is. However, if

TraceOption is 1, the thread ID is appended to the filename provided. See "TraceOption" for more information.

- TraceCategory

  Valid Values:

  - 0 = None
  - 1 = OLEDB Interface method entry
  - 2 = OLEDB Interface method exit
  - 4 = Distributed Transaction Enlistment and Delistment

  TraceCategory specifies the information that is to be traced. Combinations of different tracing categories can be made by simply adding the valid values. For example, set TraceCategory to 3 to trace all OLE DB interface method entries and exits.

- TraceLevel

  Valid Values:

  - 0 = None
  - 1 = Data Source object
  - 2 = Session object
  - 4 = Command object
  - 8 = Rowset object
  - 16 = Error object
  - 64 = Multiple Results Object

  TraceLevel specifies the OLE DB objects to be traced. Because tracing all the entry and exit calls for all the OLE DB objects can be excessive, TraceLevel is provided to limit tracing to a single or multiple OLE DB objects. To obtain tracing on multiple objects, simply add the valid values. For example, if TraceLevel is set to 12 and TraceCategory is set to 3, the trace file will only contain method entry and exit for Command and Rowset objects.

  The TraceLevel value must be set to session object (2) to trace global transaction enlistment and delistment information.

- `TraceOption`

  Valid Values:

  - `0` = Single trace file
  - `1` = Multiple trace files

  `TraceOption` specifies whether to log trace information in single or multiple files for each Thread ID. If a single trace file is specified, the filename specified in `TraceFileName` is used. If multiple trace file is requested, a Thread ID is appended to the filename provided to create a trace file for each thread.

Glossary

Component Object Model (COM)

A binary standard that enables objects to interact with other objects, regardless of the programming language that each object was written in.

consumer

A consumer is any application or tool that calls to a data source or the interfaces of provider to access data. See **provider**.

Oracle Net Services

The Oracle client/server communication software that offers transparent operation to Oracle tools or databases over any type of network protocol and operating system.

PL/SQL

Oracle Corporation's procedural language extension to SQL.

provider

A provider is an interface or set of components that provides data to a consumer. As the term is used with Oracle Provider for OLE DB, a data provider is a set of COM components that transfer data from a data source to consumer, by place the data in a tabular format when called for. See **consumer**.

stored procedure

A stored procedure is a PL/SQL block that Oracle stores in the database and can be called by name from an application.

Index