Oracle® Data Provider for .NET

Developer's Guide

10*g* Release 1 (10.1)

Part No. B10117-01

December 2003

ORACLE®

Oracle Data Provider for .NET Developer's Guide, 10*g* Release 1 (10.1)

Part No. B10117-01

Primary Author: Janis Greenberg

Contributing Authors: Riaz Ahmed, Kiminari Akiyama, Steven Caminez, Naveen Doraiswamy, Neeraj Gupta, Sinclair Hsu, Alex Keh, Arnold Poon, Chithra Ramamurthy, Ashish Shah, Martha Woo

Contents

2 Installing and Configuring

3 Features of Oracle Data Provider for .NET

4 Oracle.DataAccess.Client Namespace

Volume 2

5 Oracle.DataAccess.Types Namespace (ODP.NET Types)

Send Us Your Comments

Oracle Data Provider for .NET Developer's Guide, 10*g* Release 1 (10.1)

Part No. B10117-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. 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 about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation 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 electronic mail address (optional).

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

Preface

This document is your primary source of introductory, installation, postinstallation configuration, and usage information for Oracle Data Provider for .NET.

Oracle Data Provider for .NET is an implementation of the Microsoft ADO.NET interface.

This document describes the features of Oracle Database for Windows that apply to the Windows NT Server, Windows 2000, Windows XP, and Windows Server 2003 operating systems.

This preface contains these topics:

Audience

Oracle Data Provider for .NET Developer's Guide is intended for developers who are developing applications to access an Oracle database using Oracle Data Provider for .NET. 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 Microsoft .NET Framework classes and ADO.NET and have a working knowledge of application programing using Microsoft C#, Visual Basic, or C++.

Users 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, "Introducing Oracle Data Provider for .NET"

Provides an overview of Oracle Data Provider for .NET.

Chapter 2, "Installing and Configuring"

Describes how to install Oracle Data Provider for .NET and provides system requirements.

Read this chapter *before* installing or using Oracle Data Provider for .NET.

Chapter 3, "Features of Oracle Data Provider for .NET"

Describes provider-specific features of Oracle Data Provider for .NET, including Oracle XML Database.

Chapter 4, "Oracle.DataAccess.Client Namespace"

Describes the classes and public methods Oracle Data Provider for .NET exposes for ADO.NET programmers.

Chapter 5, "Oracle.DataAccess.Types Namespace (ODP.NET Types)"

Describes the type structures and objects provided by Oracle Data Provider for .NET.

Glossary

Defines terms used in this document.

Related Documentation

For more information, see these Oracle resources:

- *Oracle Database Installation Guide for Windows*
- *Oracle Database Release Notes for Windows*
- *Oracle Database Platform Guide for Windows*
- *Oracle Database Administrator's Guide*
- *Oracle Database Application Developer's Guide - Large Objects*
- *Oracle Database New Features*
- *Oracle Database Concepts*
- *Oracle Database Reference*
- *Oracle Database SQL Reference*
- *Oracle Net Services Administrator's Guide*
- *Oracle Net Services Reference Guide*
- *Oracle Services for Microsoft Transaction Server Developer's Guide*
- *Oracle Real Application Clusters Quick Start*
- *Oracle Database Globalization Support Guide*
- *Oracle XML DB Developer's Guide*
- *Oracle XML Developer's Kit Programmer's Guide*

For information about Oracle error messages, see *Oracle Database Error Messages*. Oracle error message documentation is available only in HTML. If you only have access to the Oracle Documentation CD, you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

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/membership/`

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/documentation/`

For additional information, see:

`http://msdn.microsoft.com/netframework`

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.	*Oracle 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.
`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]
...	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 ---------- /fs1/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fs1/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;

Convention	Meaning	Example
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.	To start the 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 following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32`

Convention	Meaning	Example
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). 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 quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to 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. For Windows NT, the default location was `C:\orant`. 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 guide follow OFA conventions. Refer to *Oracle Database Platform Guide for Windows* for additional information about OFA compliances and for information about 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 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 does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

What's New in Oracle Data Provider for .NET?

This section describes new features in Oracle Data Provider for .NET 10*g* Release 1 (10.1) and provides pointers to additional information. New features information from previous releases is also retained to help those users migrating to the current release.

The following sections describe the new features in Oracle Data Provider for .NET:

- New Features in Oracle Data Provider for .NET Release 10.1
- New Features in Oracle Data Provider for .NET Release 9.2.0.4

New Features in Oracle Data Provider for .NET Release 10.1

Oracle Data Provider for .NET release 10.1 includes the following:

- Support for Oracle Grids

  ODP.NET is grid-enabled, allowing developers to take advantage of Oracle database grid support without having to make changes to their application code.

- Support for `BINARY_FLOAT` and `BINARY_DOUBLE` datatypes in the database

  ODP.NET supports the new database native types `BINARY_FLOAT` and `BINARY_DOUBLE`

  See Also: "Datatypes BINARY_FLOAT and BINARY_DOUBLE" on page 3-20

- Support for Multiple Homes

  ODP.NET can be installed in Multiple Oracle Homes.

  In order to make multiple homes available, some of the ODP.NET files include a version number, and the use of a `HOME`*`ID`* is required.

- Support for schema-based `XMLType` in the database

  ODP.NET supports the native schema-based `XMLType`.

New Features in Oracle Data Provider for .NET Release 9.2.0.4

Oracle Data Provider for .NET release 9.2.0.4, which was released on Oracle Technology Network (OTN) included the following:

- XML support in ODP.NET.

  With XML support, ODP.NET can now:

  - Store XML data natively in the database server as the Oracle database native type, `XMLType`.
  - Access relational and object-relational data as XML data from an Oracle database instance into Microsoft .NET environment, process the XML using Microsoft .NET framework.
  - Save changes to the database server using XML data.

  See Also: "ODP.NET XML Support" on page 3-51

- Support for PL/SQL Associative Array Binding

  ODP.NET supports PL/SQL Associative Array (formerly known as PL/SQL Index-By Tables) binding.

  An application can bind an `OracleParameter`, as a PL/SQL Associative Array, to a PL/SQL stored procedure using `OracleParameter` properties.

  See Also: "PL/SQL Associative Array" on page 3-26

- Support for `InitialLOBFetchSize` property on `OracleCommand` and `OracleDataReader` objects

  See Also: "Obtaining LOB Data" on page 3-15

1

Introducing Oracle Data Provider for .NET

This chapter introduces Oracle Data Provider for .NET (ODP.NET), an implementation of a data provider for the Oracle database.

This chapter contains these topics:

Overview of Oracle Data Provider for .NET (ODP.NET)

Oracle Data Provider for .NET (ODP.NET) is an implementation of a data provider for the Oracle database.

ODP.NET uses Oracle native APIs to offer fast and reliable access to Oracle data and features from any .NET application. ODP.NET also uses and inherits classes and interfaces available in the **Microsoft .NET Framework Class Library**.

For programmers using Oracle Provider for OLE DB, ADO (ActiveX Data Objects) provides an automation layer that exposes an easy programming model. ADO.NET provides a similar programming model, but without the automation layer, for better performance. More importantly, the ADO.NET model allows native providers such as ODP.NET to expose Oracle-specific features and datatypes.

ODP.NET Assembly

`Oracle.DataAccess.dll` **assembly** provides two namespaces:

- The `Oracle.DataAccess.Client` namespace contains ODP.NET classes and enumerations.
- The `Oracle.DataAccess.Types` namespace contains the Oracle Data Provider for .NET Types (ODP.NET Types).

Oracle.DataAccess.Client Classes and Enumerations

This namespace is the Oracle Data Provider for .NET (ODP.NET).

Table 1–1 lists the client classes:

Table 1–1 Oracle.DataAccess.Client Classes

Class	Description
OracleCommand Class	An `OracleCommand` object represents a SQL command, a stored procedure, or a table name
OracleCommandBuilder Class	An `OracleCommandBuilder` object provides automatic SQL generation for the `OracleDataAdapter` when updates are made to the database
OracleConnection Class	An `OracleConnection` object represents a connection to an Oracle database

Table 1–1 Oracle.DataAccess.Client Classes(Cont.)

Class	Description
OracleDataAdapter Class	An `OracleDataAdapter` object represents a data provider object that communicates with the `DataSet`
OracleDataReader Class	An `OracleDataReader` object represents a forward-only, read-only, in-memory result set
OracleError Class	The `OracleError` object represents an error reported by an Oracle database
OracleErrorCollection Class	An `OracleErrorCollection` object represents a collection of `OracleErrors`
OracleException Class	The `OracleException` object represents an exception that is thrown when Oracle Data Provider for .NET encounters an error
OracleFailoverEventArgs Class	The `OracleFailoverEventArgs` object provides event data for the `OracleConnection.Failover` event
OracleFailoverEventHandler Delegate	The `OracleFailoverEventHandler` delegate represents the signature of the method that handles the `OracleConnection.Failover` event
OracleGlobalization Class	The `OracleGlobalization` class is used to obtain and set the Oracle globalization settings of the session, thread, and local computer (read-only)
OracleInfoMessageEventHandler Delegate	The `OracleInfoMessageEventHandler` delegate represents the signature of the method that handles the `OracleConnection.InfoMessage` event
OracleInfoMessageEventArgs Class	The `OracleInfoMessageEventArgs` object provides event data for the `OracleConnection.InfoMessage` event
OracleParameter Class	An `OracleParameter` object represents a parameter for an `OracleCommand`
OracleParameterCollection Class	An `OracleParameterCollection` object represents a collection of `OracleParameters`

Table 1–1 Oracle.DataAccess.Client Classes(Cont.)

Class	Description
OracleRowUpdatedEventArgs Class	The `OracleRowUpdatedEventArgs` object provides event data for the `OracleDataAdapter.RowUpdated` event
OracleRowUpdatedEventHandler Delegate	The `OracleRowUpdatedEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdated` event
OracleRowUpdatingEventArgs Class	The `OracleRowUpdatingEventArgs` object provides event data for the `OracleDataAdapter.RowUpdating` event
OracleRowUpdatingEventHandler Delegate	The `OracleRowUpdatingEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdating` event
OracleTransaction Class	An `OracleTransaction` object represents a local transaction
OracleXmlQueryProperties Class	An `OracleXmlQueryProperties` object represents the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Query`
OracleXmlSaveProperties Class	An `OracleXmlSaveProperties` object represents the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Insert`, `Update`, or `Delete`

Table 1–2 lists the client enumerations:

Table 1–2 Oracle.DataAccess.Client Enumerations

Enumeration	Description
FailoverEvent Enumeration	`FailoverEvent` enumerated values are used to explicitly specify the state of the failover
FailoverReturnCode Enumeration	`FailoverReturnCode` enumerated values are passed back by the application to the ODP.NET provider to request a retry in case of a failover error or to continue in case of a successful failover
FailoverType Enumeration	`FailoverType` enumerated values are used to indicate the type of failover event that was raised
OracleDbType Enumeration	`OracleDbType` enumerated values are used to explicitly specify the `OracleDbType` of an `OracleParameter`
OracleParameterStatus Enumeration	The `OracleParameterStatus` enumeration type indicates whether a `NULL` value is fetched from a column, whether truncation has occurred during the fetch, or whether a `NULL` value is to be inserted into a database column
OracleXmlCommandType Enumeration	The `OracleXmlCommandType` enumeration specifies the values that are allowed for the `OracleXmlCommandType` property of the `OracleCommand` class

Oracle.DataAccess.Types Classes and Structures

The `Oracle.DataAccess.Types` namespace provides classes and structures for Oracle native types that can be used with Oracle Data Provider for .NET.

Table 1–3 lists the types structures:

Table 1–3 Oracle.DataAccess.Types Structures

Structure	Description
OracleBinary Structure	The `OracleBinary` structure represents a variable-length stream of binary data
OracleDate Structure	The `OracleDate` structure represents the Oracle `DATE` datatype
OracleDecimal Structure	The `OracleDecimal` structure represents an Oracle `NUMBER` in the database or any Oracle numeric value
OracleIntervalDS Structure	The `OracleIntervalDS` structure represents the Oracle `INTERVAL DAY TO SECOND` datatype
OracleIntervalYM Structure	The `OracleIntervalYM` structure represents the Oracle `INTERVAL YEAR TO MONTH` datatype
OracleString Structure	The `OracleString` structure represents a variable-length stream of characters
OracleTimeStamp Structure	The `OracleTimeStamp` structure represents the Oracle `TimeStamp` datatype
OracleTimeStampLTZ Structure	The `OracleTimeStampLTZ` structure represents the Oracle `TIMESTAMP WITH LOCAL TIME ZONE` data type
OracleTimeStampTZ Structure	The `OracleTimeStampTZ` structure represents the Oracle `TIMESTAMP WITH TIME ZONE` data type

Type Exceptions are thrown only by ODP.NET type structures. Table 1–4 lists the type exceptions:

Table 1–4 Oracle.DataAccess.Types Exceptions

Exception	Description
OracleTypeException Class	The `OracleTypeException` object is the base exception class for handling exceptions that occur in the ODP.NET Type classes
OracleNullValueException Class	The `OracleNullValueException` represents an exception that is thrown when trying to access an ODP.NET Type structure that is `null`
OracleTruncateException Class	The `OracleTruncateException` class represents an exception that is thrown when truncation in an ODP.NET Type class occurs

Table 1–5 lists the types classes:

Table 1–5 Oracle.DataAccess.Types Classes

Class	Description
OracleBFile Class	An `OracleBFile` is an object that has a reference to `BFILE` data. It provides methods for performing operations on `BFiles`
OracleBlob Class	An `OracleBlob` object is an object that has a reference to `BLOB` data. It provides methods for performing operations on `BLOBs`
OracleClob Class	An `OracleClob` is an object that has a reference to `CLOB` data. It provides methods for performing operations on `CLOBs`
OracleRefCursor Class	An `OracleRefCursor object represents an Oracle REF CURSOR`
OracleXmlStream Class	An `OracleXmlStream` object represents a sequential read-only stream of XML data stored in an `OracleXmlType` object
OracleXmlType Class	An `OracleXmlType` object represents an Oracle `XmlType` instance

Using ODP.NET in a Simple Application

The following is a very simple C# application that connects to an Oracle database and displays its version number before disconnecting.

```
using System;
using Oracle.DataAccess.Client;

class Example
{
  OracleConnection con;

  void Connect()
  {
    con = new OracleConnection();
    con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle";
    con.Open();
    Console.WriteLine("Connected to Oracle" + con.ServerVersion);
  }

  void Close()
  {
    con.Close();
    con.Dispose();
  }

  static void Main()
  {
    Example example = new Example();
    example.Connect();
    example.Close();
  }
}
```

Note: Additional samples are provided in the *ORACLE_BASE**ORACLE_HOME*\ODP.NET\Samples directory.

2

Installing and Configuring

This chapter describes installation and configuration requirements for Oracle Data Provider for .NET.

This chapter contains these topics:

System Requirements

Oracle Data Provider for .NET requires the following:

- Microsoft .NET Framework 1.0 or higher.
- Windows NT, Windows XP, Windows 2000, or Windows Server 2003.
- Access to Oracle8*i* Database release 3 (8.1.7) or higher.
- Oracle Client release 10.1 or higher and Net Services (included with ODP.NET Software).

Additional requirements:

- Applications using Microsoft Enterprise Services transactions require Oracle Services for Microsoft Transaction Server release 10.1.
- `OracleXmlStream` and `OracleXmlType` classes require Oracle9*i* Database release 2 (9.2) or higher.
- Applications using `OracleXmlStream` and `OracleXmlType` classes with schema-based XMLType require Oracle Database 10*g*.
- For database releases 8.1.7 and 9.0.1 only: To provide XML support, the following `OracleCommand` methods, require Oracle XML Developer's Kit (Oracle XDK) release 9.2 or higher to be installed on the database. Oracle XDK can be downloaded from Oracle Technology Network (OTN).
  - `ExecuteStream`
  - `ExecuteToStream`
  - `ExecuteXmlReader`
  - `ExecuteNonQuery`

  See Also:

  - http://msdn.microsoft.com/netframework
  - http://otn.oracle.com/tech/xml/xdkhome.html to download the XDK

Installing Oracle Data Provider for .NET

When you install Oracle Data Provider for .NET, Oracle Universal Installer automatically registers ODP.NET with the Global Assembly Cache (GAC).

Additionally, ODP.NET Integrated Help is registered with Visual Studio .Net.

> **See Also:** *Oracle Database Installation Guide for Windows* for installation instructions

File Locations

The `Oracle.DataAccess.dll` assembly is installed in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\bin` directory.

Documentation and the `readme.txt` are installed in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\ODP.NET\doc` directory.

Samples are provided in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\ODP.NET\Samples` directory.

3

Features of Oracle Data Provider for .NET

This chapter describes Oracle Data Provider for .NET provider-specific features and how to use them to develop .NET applications.

This chapter contains these topics:

Connecting to the Oracle Database

This section describes `OracleConnection` provider-specific features, including:

- Connection String Attributes
- Connection Pooling
- Operating System Authentication
- Privileged Connections
- Password Expiration
- Proxy Authentication
- Transparent Application Failover (TAF) Callback Support

Connection String Attributes

Table 3–1 lists the supported connection string attributes.

Table 3–1 Supported Connection String Attributes

Connection String Attribute	Default value	Description
`Connection Lifetime`	`0`	Maximum life time (in seconds) of the connection
`Connection Timeout`	`15`	Maximum time (in seconds) to wait for a free connection from the pool
`Data Source`	empty string	Oracle Net Service Name that identifies the database to connect to
`DBA Privilege`	empty string	Administrative privileges: `SYSDBA` or `SYSOPER`
`Decr Pool Size`	1	Controls the number of connections that are closed when an excessive amount of established connections are unused
`Enlist`	`true`	Enables or disables serviced components to automatically enlist in distributed transactions
`Incr Pool Size`	5	Controls the number of connections that are established when all the connections in the pool are used
`Max Pool Size`	`100`	Maximum number of connections in a pool

Table 3–1 Supported Connection String Attributes (Cont.)

Connection String Attribute	Default value	Description
`Min Pool Size`	`1`	Minimum number of connections in a pool
`Password`	empty string	Password for the user specified by `User Id`
`Persist Security Info`	`false`	Enables or disables the retrieval of password in the connection string
`Pooling`	`true`	Enables or disables connection pooling
`Proxy User Id`	empty string	User name of the proxy user
`Proxy Password`	empty string	Password of the proxy user
`User Id`	empty string	Oracle user name

The following example uses connection string attributes to connect to an Oracle Database:

```
// C#
...
OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
con.Open();
...
```

> **See Also:** "OracleConnection Properties" on page 4-60 for detailed information on connection attributes

Connection Pooling

ODP.NET connection pooling is enabled and disabled using the `Pooling` connection string attribute. By default, connection pooling is enabled. The following are `ConnectionString` attributes that control the behavior of the connection pooling service:

- `Pooling`
- `Connection Lifetime`
- `Connection Timeout`
- `Max Pool Size`

- Min Pool Size
- Incr Pool Size
- Decr Pool Size

Connection Pooling Example

The following code opens a connection using ConnectionString attributes related to connection pooling.

```
// C#
...
OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;" +
         "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
         "Incr Pool Size=5; Decr Pool Size=2";
con.Open();
...
```

With connection pooling enabled (the default), the Open and Close methods of the OracleConnection object implicitly use the connection pooling service. In the preceding code, the Open call uses the connection pooling service, which is responsible for returning a connection to the application.

Connection pools are created by the connection pooling service using the ConnectionString as a signature to uniquely identify a pool.

If no pool with the exact attribute values in the ConnectionString exists, the connection pooling service creates a new connection pool. If a pool already exists with the requested signature, a connection is returned to the application from that pool.

When a connection pool is created, the connection-pooling service initially creates the number of connections defined by the Min Pool Size attribute of the ConnectionString. This number of connections is always maintained by the connection pooling service for the connection pool.

At any given time, these connections are available in the pool or used by the application.

The Incr Pool Size attribute of the ConnectionString defines the number of new connections to be created by the connection pooling service when more connections are needed in the connection pool.

When the application closes a connection, the connection pooling service determines whether the connection lifetime has exceeded the Connection

Lifetime attribute; if so, the connection pooling service closes the connection; otherwise, the connection goes back to the connection pool. The connection pooling service only enforces the Connection Lifetime when a connection is going back to the connection pool.

The Max Pool Size attribute of the ConnectionString sets the maximum number of connections for a connection pool. If a new connection is requested, no connections are available, and Max Pool Size has been reached, then the connection pooling service waits for the time defined by Connection Timeout. If the Connection Timeout has been reached and there are still no connections available in the pool, the connection pooling service raises an exception indicating that the pooled connection request has timed-out.

The connection pooling service closes connections when they are not used; connections are closed every three minutes. The Decr Pool Size attribute of the ConnectionString provides connection pooling service for the maximum number of connections that can be closed in one run.

Operating System Authentication

The Oracle Database can use Windows user login credentials to authenticate database users. To open a connection using Windows user login credentials, the User Id ConnectionString attribute must be set to /. If Password is provided, it is ignored.

```
// C#
...
OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=/;Data Source=oracle;";
con.Open();
...
```

> **See Also:** *Oracle Database Platform Guide for Windows* for information on how to set up an Oracle Database to authenticate database users using Windows user login credentials

Privileged Connections

Oracle allows database administrators to connect to an Oracle Database with either SYSDBA or SYSOPER privileges. This is done through the DBA Privilege attribute of the ConnectionString.

The following example connects SYS/SYS as SYSDBA:

```
// C#
...
OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=SYS;Password=SYS;" +
    "DBA Privilege=SYSDBA;Data Source=oracle;";
con.Open();
...
```

See Also: "DBA Privilege" on page 4-63 for further information on privileged connections in the database server

Password Expiration

Oracle allows users' password to expire. ODP.NET lets applications handle the password expiration by providing a new method, OpenWithNewPassword, that opens the connection with a new password.

The following code snippet uses the OracleConnection OpenWithNewPassword method to connect with a new password of panther:

```
// C#
...
OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
//  Here the con.Open() fails if the password has expired.
//  An application catches this and attempts to reconnect with a new password
//  of "panther". The initial connection string must remain intact.

try {
con.Open();
}

catch {
con.OpenWithNewPassword("panther");
}

...
```

Note: OpenWithNewPassword should only be used when the user password has expired, not for changing the password.

See Also: "OpenWithNewPassword" on page 4-79 for further information on connecting after the password expires

Proxy Authentication

The Oracle Database allows a middle-tier server to connect to proxy clients in a secure fashion.

In multitier environments, proxy authentication allows control of middle-tier application security by preserving client identities and privileges through all tiers, and by auditing actions taken on behalf of clients. The proxy authentication feature allows the identity of a user using a Web application to be passed through the application to the database server.

ODP.NET supports proxy authentication with or without a client password by providing the `Proxy User Id` and `Proxy Password` attributes of the `ConnectionString` property.

```
// C#
...
OracleConnection con = new OracleConnection();
// Connecting using proxy authentication
con.ConnectionString = "User Id=customer;Password=lion;" +
     "Data Source=oracle;Proxy User Id=appserver;Proxy Password=eagle; ";
con.Open();
...
```

See Also:

- *Oracle Database Application Developer's Guide - Fundamentals* for details on designing a middle-tier server using proxy users
- *Oracle Database SQL Reference* for the description and syntax of the proxy clause for `ALTER USER`
- *Oracle Database Administrator's Guide* section "Auditing in a Multi-Tier Environment"

Transparent Application Failover (TAF) Callback Support

Transparent Application Failover (TAF) is a feature in the Oracle Database that provides high availability.

TAF enables an application connection to automatically reconnect to a database if the connection fails. Active transactions roll back, but the new database connection,

made by way of a different node, is identical to the original. This is true regardless of how the connection fails.

With Transparent Application Failover, a client notices no loss of connection as long as there is one instance left serving the application. The database administrator controls which applications run on which instances and also creates a failover order for each application.

Given the delays that failovers can cause, applications may wish to be notified by a TAF callback. ODP.NET supports TAF callback through the `Failover` event of the `OracleConnection` object, which allows applications to be notified whenever a failover occurs. To receive TAF callbacks, an event handler function must be registered with the `Failover` event.

When a failover occurs, the `Failover` event is raised and the registered event handler is invoked several times during the course of reestablishing the connection to another Oracle instance.

The first call to the event handler occurs when the Oracle Database first detects an instance connection loss. This allows the application to act accordingly for the upcoming delay for the failover.

If the failover is successful, the `Failover` event is raised again when the connection is reestablished and usable. At this time, the application can resynchronize the `OracleGlobalization` session setting and inform the application user that a failover has occurred.

If failover is unsuccessful, the `Failover` event is raised to inform the application that a failover did not take place.

The application can determine whether or not the failover is successful by checking the `OracleFailoverEventArgs` that is passed to the event handler.

The following code example registers an event handler method called `OnFailover`:

```
// C#
...
OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
con.Open();
con.Failover += new OracleFailoverEventHandler(OnFailover);
...
```

The `Failover` event only invokes one event handler. If multiple `Failover` event handlers are registered with the `Failover` event, only the event handler registered last is invoked.

Note: Distributed transactions are not supported in an environment where failover is enabled.

See Also:

- *Oracle Net Services Administrator's Guide*
- "OracleFailoverEventHandler Delegate" on page 4-209
- "OracleFailoverEventArgs Class" on page 4-204

ODP.NET Types Overview

ODP.NET types represent Oracle native types as a structure or as a class. ODP.NET type structures follow **value semantics** while ODP.NET type classes follow **reference semantics**. ODP.NET types provide safer and more efficient ways of obtaining Oracle native data in a .NET application than .NET types. For example, an `OracleDecimal` holds up to 38 precisions while a .NET `Decimal` holds only up to 28 precisions.

Table 3–2 lists all the Oracle native types supported by ODP.NET and their corresponding ODP.NET type. The third column lists the .NET Framework datatype that corresponds to the `Value` property of each ODP.NET Type.

Table 3–2 Oracle Native Types Supported by ODP.NET

Oracle Native Type	ODP.NET Type	.NET Framework Datatypes
`BFILE`	`OracleBFile` class	`System.Byte[]`
`BLOB`	`OracleBlob` class	`System.Byte[]`
`CHAR`	`OracleString` structure	`System.String`
`CLOB`	`OracleClob` class	`System.String`
`DATE`	`OracleDate` structure	`System.DateTime`
`INTERVAL DAY TO SECOND`	`OracleIntervalDS` structure	`System.TimeSpan`

Table 3–2 Oracle Native Types Supported by ODP.NET (Cont.)

Oracle Native Type	ODP.NET Type	.NET Framework Datatypes
INTERVAL YEAR TO MONTH	OracleIntervalYM structure	System.Int64
LONG	OracleString structure	System.String
LONG RAW	OracleBinary structure	System.Byte[]
NCLOB	OracleClob class	System.String
NCHAR	OracleString structure	System.String
NUMBER	OracleDecimal structure	System.Decimal
NVARCHAR2	OracleString structure	System.String
RAW	OracleBinary structure	System.Byte[]
REF CURSOR	OracleRefCursor class	Not Applicable
TIMESTAMP	OracleTimeStamp structure	System.DateTime
TIMESTAMP WITH LOCAL TIME ZONE	OracleTimeStampLTZ structure	System.DateTime
TIMESTAMP WITH TIME ZONE	OracleTimeStampTZ structure	System.DateTime
UROWID	OracleString structure	System.String
VARCHAR2	OracleString structure	System.String
XMLType	OracleXmlType class	System.String

Obtaining Data From an OracleDataReader

The `ExecuteReader` method of the `OracleCommand` object returns an `OracleDataReader` object, which is a read-only, forward-only result set.

This section provides the following information about the `OracleDataReader`:

- Typed OracleDataReader Accessors
- Obtaining LONG and LONG RAW Data
- Obtaining LOB Data
- Controlling the Number of Rows Fetched in One Server Round-Trip

Typed OracleDataReader Accessors

The `OracleDataReader` provides two types of typed accessors:

- .NET Type Accessors
- ODP.NET Type Accessors

.NET Type Accessors

Table 3–3 lists all the Oracle native database types that ODP.NET supports and the corresponding .NET Type that best represents the Oracle native type. The third column indicates the valid typed accessor that can be invoked for an Oracle native type to be obtained as a .NET type. If an invalid typed accessor is used for a column, an `InvalidCastException` is thrown. Oracle native datatypes depend on the version of the database; therefore, some datatypes are not available in earlier versions of Oracle Database.

> **See Also:** "OracleDataAdapter Class" on page 4-86 for more information

Table 3–3 .NET Type Accessors

Oracle Native Datatype	.NET Type	Typed Accessor
`BFILE`	`System.Byte[]`	`GetBytes`
`BINARY_DOUBLE`	`System.Double`	`GetDouble`
`BINARY_FLOAT`	`System.Single`	`GetFloat`
`BLOB`	`System.Byte[]`	`GetBytes`

Table 3–3 .NET Type Accessors (Cont.)

Oracle Native Datatype	.NET Type	Typed Accessor
CHAR	System.String	GetString GetChars
CLOB	System.String	GetString GetChars
DATE	System.DateTime	GetDateTime
INTERVAL(DS)	System.Interval	GetTimeSpan
INTERVAL (YM)	System.Interval	GetTimeSpan
LONG	System.String	GetString GetChars
LONG RAW	System.Byte[]	GetBytes
NCHAR	System.String	GetString GetChars
NCLOB	System.String	GetString GetChars
NUMBER	System.Decimal	GetDecimal
NVARCHAR2	System.String	GetString GetChars
RAW	System.Byte[]	GetBytes
ROWID	System.String	GetString GetChars
TIMESTAMP	System.TimeStamp	GetTimeStamp
TIMESTAMP WITH LOCAL TIME ZONE	System.TimeStamp	GetTimeStamp
TIMESTAMP WITH TIME ZONE	System.TimeStamp	GetTimeStamp
UROWID	System.String	GetString GetChars
VARCHAR2	System.String	GetString GetChars

Table 3–3 .NET Type Accessors (Cont.)

Oracle Native Datatype	.NET Type	Typed Accessor
XMLType	System.String	GetString
	System.Xml.XmlReader	GetXmlReader

ODP.NET Type Accessors

ODP.NET exposes provider-specific types that natively represent the datatypes in the database. In some cases, these ODP.NET types provide better performance and functionality that is not available to the corresponding .NET types. The ODP.NET types can be obtained from the `OracleDataReader` by calling their respective typed accessor.

> **See Also:** "ODP.NET Types Overview" on page 3-9 for a list of all ODP.NET types

Table 3–4 lists the valid type accessors that ODP.NET uses to obtain ODP.NET Types for an Oracle native type.

Table 3–4 ODP.NET Type Accessors

Oracle Native Database Type	ODP.NET Type	Typed Accessor
BFILE	OracleBFile	GetOracleBFile
BLOB	OracleBlob	GetOracleBlob
	OracleBlob	GetOracleBlobForUpdate
	OracleBinary	GetOracleBinary
CHAR	OracleString	GetOracleString
CLOB	OracleClob	GetOracleClob
	OracleClob	GetOracleClobForUpdate
	OracleString	GetOracleString
DATE	OracleDate	GetOracleDate
INTERVAL (DS)	OracleIntervalDS	GetOracleIntervalDS
INTERVAL (YM)	OracleIntervalYM	GetOracleIntervalYM
LONG	OracleString	GetOracleString
LONG RAW	OracleBinary	GetOracleBinary

Table 3–4 ODP.NET Type Accessors (Cont.)

Oracle Native Database Type	ODP.NET Type	Typed Accessor
NCHAR	OracleString	GetOracleString
NCLOB	OracleString	GetOracleString
NUMBER	OracleDecimal	GetOracleDecimal
NVARCHAR2	OracleString	GetOracleString
RAW	OracleBinary	GetOracleBinary
ROWID	OracleString	GetOracleString
TIMESTAMP	OracleTimeStamp	GetOracleTimeStamp
TIMESTAMP WITH LOCAL TIME ZONE	OracleTimeStampLTZ	GetOracleTimeStampLTZ
TIMESTAMP WITH TIME ZONE	OracleTimeStampTZ	GetOracleTimeStampTZ
UROWID	OracleString	GetOracleString
VARCHAR2	OracleString	GetOracleString
XMLType	OracleString	GetOracleString
	OracleXmlType	GetOracleXmlType

Obtaining LONG and LONG RAW Data

When an OracleDataReader is created containing a LONG or LONG RAW column type, OracleDataReader determines whether this column data needs to be fetched immediately or not, by checking the value of the InitialLONGFetchSize property of the OracleCommand that created the OracleDataReader.

By default, InitialLONGFetchSize is set to 0. If the InitialLONGFetchSize property value of the OracleCommand is left as 0, the entire LONG or LONG RAW data retrieval is deferred until that data is explicitly requested by the application. If the InitialLONGFetchSize property is set to a nonzero value, the LONG or LONG RAW data is immediately fetched up to the number of characters or bytes that the InitialLONGFetchSize property specifies.

ODP.NET does not support CommandBehavior.SequentialAccess. Therefore, LONG and LONG RAW data can be fetched in a random fashion.

To obtain data beyond InitialLONGFetchSize bytes or characters, one of the following must be in the select list:

- primary key
- ROWID
- unique columns - (defined as a set of columns on which a unique constraint has been defined or a unique index has been created, where at least one of the columns in the set has a NOT NULL constraint defined on it)

The requested data is fetched from the database when the appropriate typed accessor method (GetOracleString or GetString for LONG or GetOracleBinary or GetBytes for LONG RAW) is called on the OracleDataReader object.

In order to fetch the data in a non-defer mode or when the columns in the select list do not have a primary key column, a ROWID, or unique columns, set the size of the InitialLONGFetchSize property on the OracleCommand object to equal or greater than the amount of bytes or characters needed to be retrieved.

Obtaining LOB Data

When an OracleDataReader is created containing LOB column types, OracleDataReader determines whether the LOB column data needs to be fetched immediately or not by checking the value of the InitialLONGFetchSize property of the OracleCommand that created the OracleDataReader. By default, InitialLOBFetchSize is set to 0. If the InitialLOBFetchSize property value of the OracleCommand is left as 0, the entire LOB data retrieval is deferred until that data is explicitly requested by the application. If the InitialLOBFetchSize property is set to a nonzero value, the LOB data is immediately fetched up to the number of characters or bytes that the InitialLOBFetchSize property specifies.

By default, when InitialLOBFetchSize property is 0, GetOracleBlob() and GetOracleClob() can be invoked on the OracleDataReader to obtain OracleBlob and OracleClob objects. However, if the InitialLOBFetchSize is set to a nonzero value, GetOracleBlob() and GetOracleClob() methods are disabled. In this scenario, the BLOB and CLOB data needs to be fetched by using GetBytes() and GetChars(), respectively.

Methods Supported or Not Supported for InitialLOBFetchSize

Table 3–5 and Table 3–6 list supported and not supported methods for the CLOB and BLOB datatypes when the OracleCommand InitialLOBFetchSize property is set to a nonzero value.

Table 3–5 OracleDataReader CLOB Methods

Supported	Not Supported
GetChars	GetOracleClob
GetString	GetOracleClobForUpdate
GetValue	GetOracleValue
GetValues	GetOracleValues
GetOracleString	

Table 3–6 OracleDataReader BLOB Methods

Supported	Not Supported
GetBytes	GetOracleBlob
GetValue	GetOracleBlobForUpdate
GetValues	GetOracleValue
GetOracleBinary	GetOracleValues

LOB Data Fetching Considerations

ODP.NET does not support `CommandBehavior.SequentialAccess`. Therefore, LOB data can be fetched in a random fashion.

To obtain data beyond `InitialLOBFetchSize` bytes or characters, one of the following must be in the select list:

- primary key
- `ROWID`
- unique columns - (defined as a set of columns on which a unique constraint has been defined or a unique index has been created, where at least one of the columns in the set has a `NOT NULL` constraint defined on it)

The requested data is fetched from the database when the appropriate typed accessor method is called on the `OracleDataReader` object. Note that the primary key column is not required if `InitialLOBFetchSize` is set to `0`.

In order to fetch the data in a non-defer mode or when the columns in the select list do not have a primary key column, a `ROWID`, or unique columns, set the size of the `InitialLOBFetchSize` property on the `OracleCommand` object to an amount equal to or greater than the bytes or characters that need to be retrieved.

Performance

Setting `InitialLOBFetchSize` to a nonzero value can improve performance in certain cases. Using `InitialLOBFetchSize` can provide better performance than retrieving the underlying LOB data using `OracleBlob` or `OracleClob` objects. This is true if an application does not need to obtain `OracleBlob` and `OracleClob` objects from the `OracleDataReader` and the size the LOB column data is not very large. `InitialLOBFetchSize` is particularly useful in cases where the size of the LOB column data returned by query is approximately the same for all the rows.

It is generally recommended that `InitialLOBFetchSize` be set to a value larger than the size of the LOB data for more than 80% of the rows returned by the query. For example, if the size of the LOB data is less than 1 KB in 80% of the rows and more than 1 MB for 20% of the rows, set `InitialLOBFetchSize` to 1 KB.

See Also:

- "LOB Support" on page 3-35
- "InitialLOBFetchSize" on page 4-20
- "InitialLONGFetchSize" on page 4-21

Controlling the Number of Rows Fetched in One Server Round-Trip

Application performance depends on the number of rows the application needs to fetch and the number of database round-trips that are needed to retrieve them.

Use of FetchSize

The `FetchSize` property represents the total memory size in bytes that ODP.NET allocates to cache the data fetched from a server round-trip.

The `FetchSize` property can be set either on the `OracleCommand` or the `OracleDataReader` depending on the situation. Additionally, the `FetchSize` property of the `OracleCommand` is inherited by the `OracleDataReader` and can be modified.

If the `FetchSize` property is set on the `OracleCommand`, then the newly created `OracleDataReader` inherits the `FetchSize` property of the `OracleCommand`. This inherited `FetchSize` can be left as is or modified to override the inherited value. The `FetchSize` property of the `OracleDataReader` object can be changed before the first `Read` method invocation, which allocates memory specified by the `FetchSize`. All subsequent fetches from the database use the same cache allocated

for that OracleDataReader. Therefore, changing the FetchSize after the first Read method invocation has no effect.

Fine-Tuning FetchSize

By fine-tuning the FetchSize property, applications can control memory usage and the number of rows fetched in one server round-trip for better performance. For example, if a query returns 100 rows and each row takes 1024 bytes, then setting FetchSize to 102400 takes just one server round-trip to fetch the hundred rows. For the same query, if the FetchSize is set to 10240, it takes 10 server round-trips to retrieve 100 rows. If the application requires all the rows to be fetched from the result set, the first scenario is faster than the second. However, if the application requires just the first 10 rows from the result set, the second scenario can perform better since it only fetches 10 rows and not 100 rows.

Using the RowSize Property

The RowSize property of the OracleCommand object is populated with the row size (in bytes) after an execution of a SELECT statement. The FetchSize property can then be set to a value relative to the RowSize by setting it to the product of RowSize and the number of rows to fetch for each server round-trip.

For example, setting the FetchSize to RowSize * 10 forces the OracleDataReader to fetch exactly 10 rows for each server round-trip. Note that the RowSize does not change due to the data length in each individual columns. Instead, the RowSize is determined strictly from the metadata information of the database table(s) that the SELECT is executed against.

The RowSize property can be used to set the FetchSize at design time or at runtime as described in the following sections.

Setting FetchSize Value at Design Time If the row size for a particular SELECT statement is already known from a previous execution, FetchSize of the OracleCommand can be set at design time to the product of that row size and the number of rows the application wishes to fetch for each server round-trip. The FetchSize value set on the OracleCommand object is inherited by the OracleDataReader that is created by the ExecuteReader method invocation on the OracleCommand. Rather than setting the FetchSize on the OracleCommand, the FetchSize can also be set on the OracleDataReader directly. In either case, the FetchSize is set at design time without accessing the RowSize property value at runtime.

Setting FetchSize Value at Runtime Applications that do not know the row size at design time can use the `RowSize` property of the `OracleCommand` object to set the `FetchSize` property of the `OracleDataReader` object. The `RowSize` property provides a dynamic way of setting the `FetchSize` property based on the size of a row.

After an `OracleDataReader` object is obtained by invoking the `ExecuteReader` method on the `OracleCommand`, the `RowSize` property is populated with the size of the row (in bytes). By using the `RowSize` property, the application can dynamically set the `FetchSize` property of the `OracleDataReader` to the product of the `RowSize` property value and the number of rows the application wishes to fetch for each server round-trip. In this scenario, the `FetchSize` is set by accessing the `RowSize` property at runtime.

OracleCommand Object

The `OracleCommand` object represents SQL statements or stored procedures executed on the Oracle Database.

This section includes the following topics:

- Transaction
- Parameter Binding

Transaction

The Oracle Database starts a transaction only in the context of a connection. Once a transaction starts, all the successive command execution on that connection run in the context of that transaction. Transactions can only be started on a `OracleConnection` object and the read-only `Transaction` property on the `OracleCommand` object is implicitly set by the `OracleConnection` object. Therefore, the application cannot set the `Transaction` property, nor does it need to.

Parameter Binding

ODP.NET allows applications to retrieve data as either a .NET Framework type or an ODP.NET type.

How the data is retrieved depends on whether application sets the `OUT` parameter to the `DbType` property (.NET type) or `OracleDbType` property (ODP.NET type) of the `OracleParameter`.

For example, if the output parameter is bound as an `DbType.String`, the output data is returned as a .NET `String`. On the other hand, if the parameter is bound as `OracleDbType.Char`, the output data is returned as `OracleString` type.

When the `DbType` of an `OracleParameter` is set, the `OracleDbType` of the `OracleParameter` changes accordingly, and vice versa. The parameter set last prevails.

Lastly, an application can simply bind the data and have ODP.NET infer both the `DbType` and `OracleDbType` from the .NET type of the parameter value.

ODP.NET populates `InputOutput`, `Output`, and `ReturnValue` parameters with the Oracle data, through the execution of the following `OracleCommand` methods:

- `ExecuteReader`
- `ExecuteNonQuery`
- `ExecuteScalar`

An application should not bind a value for output parameters; it is the responsibility of ODP.NET to create the value object and populate the `OracleParameter Value` property with the object.

This section describes the following:

- Datatypes BINARY_FLOAT and BINARY_DOUBLE
- OracleDbType Enumeration Type
- Inference of DbType, OracleDbType, and .NET Types
- PL/SQL Associative Array
- Array Binding

See Also: "OracleDbType Enumeration" on page 4-361

Datatypes BINARY_FLOAT and BINARY_DOUBLE

Starting from Oracle Database 10*g*, the database supports two new native datatypes, `BINARY_FLOAT` and `BINARY_DOUBLE`.

`BINARY_FLOAT` and `BINARY_DOUBLE` datatypes represent single-precision and double-precision floating point values respectively.

In `OracleParameter` binding, an application should use the enumerations `OracleDbType.Float` and `OracleDbType.Double` for `BINARY_FLOAT` and `BINARY_DOUBLE` datatypes.

See Also:

- "GetDouble" on page 4-139
- "GetFloat" on page 4-141

OracleDbType Enumeration Type

`OracleDbType` enumerated values are used to explicitly specify the `OracleDbType` of an `OracleParameter`.

Table 3–7 lists all the `OracleDbType` enumeration values with a description of each enumerated value.

Table 3–7 OracleDbType Enumeration Values

Member Name	Description
`BFile`	Oracle `BFILE` type
`Blob`	Oracle `BLOB` type
`Byte`	`byte` type
`Char`	Oracle `CHAR` type
`Clob`	Oracle `CLOB` type
`Date`	Oracle `DATE` type
`Decimal`	Oracle `NUMBER` type
`Double`	8-byte `FLOAT` type
`Int16`	2-byte `INTEGER` type
`Int32`	4-byte `INTEGER` type
`Int64`	8-byte `INTEGER` type
`IntervalDS`	Oracle `INTERVAL DAY TO SECOND` type
`IntervalYM`	Oracle `INTERVAL YEAR TO MONTH` type
`Long`	Oracle `LONG` type
`LongRaw`	Oracle `LONG RAW` type
`NChar`	Oracle `NCHAR` type
`NClob`	Oracle `NCLOB` type
`NVarchar2`	Oracle `NVARCHAR2` type

Table 3–7 OracleDbType Enumeration Values (Cont.)

Member Name	Description
Raw	Oracle RAW type
RefCursor	Oracle REF CURSOR type
Single	4-byte FLOAT type
TimeStamp	Oracle TIMESTAMP type
TimeStampLTZ	Oracle TIMESTAMP WITH LOCAL TIME ZONE type
TimeStampTZ	Oracle TIMESTAMP WITH TIME ZONE type
Varchar2	Oracle VARCHAR2 type
XmlType	Oracle XMLType type

Inference of DbType, OracleDbType, and .NET Types

This section explains the inference from the System.Data.DbType, OracleDbType, and Value properties in the OracleParameter class.

In the OracleParameter class, DbType, OracleDbType, and Value properties are linked. Specifying the value of any of these properties infers the value of one or more of the other properties.

Inference of DbType from OracleDbType In the OracleParameter class, specifying the value of OracleDbType infers the value of DbType as shown in Table 3–8.

Table 3–8 Inference of System.Data.DbType from OracleDbType

OracleDbType	System.Data.DbType
BFile	Object
Blob	Object
Byte	Byte
Char	StringFixedLength
Clob	Object
Date	Date
Decimal	Decimal
Double	Double
Int16	Int16

Table 3–8 Inference of System.Data.DbType from OracleDbType (Cont.)

OracleDbType	System.Data.DbType
Int32	Int32
Int64	Int64
IntervalDS	TimeSpan
IntervalYM	Int64
Long	String
LongRaw	Binary
NChar	StringFixedLength
NClob	Object
NVarchar2	String
Raw	Binary
RefCursor	Object
Single	Single
TimeStamp	DateTime
TimeStampLTZ	DateTime
TimeStampTZ	DateTime
Varchar2	String
XmlType	String

Inference of OracleDbType from DbType In the `OracleParameter` class, specifying the value of `DbType` infers the value of `OracleDbType` as shown in Table 3–9.

Table 3–9 Inference of OracleDbType from DbType

System.Data.DbType	OracleDbType
Binary	Raw
Boolean	*Not Supported*
Byte	Byte
Currency	*Not Supported*
Date	Date

Table 3–9 Inference of OracleDbType from DbType (Cont.)

System.Data.DbType	OracleDbType
DateTime	TimeStamp
Decimal	Decimal
Double	Double
Guid	*Not Supported*
Int16	Int16
Int32	Int32
Int64	Int64
Object	*Not Supported*
Sbyte	*Not Supported*
Single	Single
String	Varchar2
StringFixedLength	Char
Time	TimeStamp
UInt16	*Not Supported*
UInt32	*Not Supported*
Uint64	*Not Supported*
VarNumeric	*Not Supported*

Inference of DbType and OracleDbType from Value In the `OracleParameter` class, `Value` is an object type which can be of any .NET Framework datatype or ODP.NET type. If the `OracleDbType` and `DbType` in the `OracleParameter` object are not specified, `OracleDbType` is inferred from the type of the `Value` property.

Table 3–10 shows the inference of DbType and OracleDbType from Value when the type of Value is one of the .NET Framework datatypes.

Table 3–10 Inference of DbType and OracleDbType from Value (.NET Datatypes)

Value (.NET Datatypes)	System.Data.DbType	OracleDbType
Byte	Byte	Byte
Byte[]	Binary	Raw
Char / Char []	String	Varchar2
DateTime	DateTime	TimeStamp
Decimal	Decimal	Decimal
Double	Double	Double
Float	Single	Single
Int16	Int16	Int16
Int32	Int32	Int32
Int64	Int64	Int64
Single	Single	Single
String	String	Varchar2
TimeSpan	TimeSpan	IntervalDS

Note: Using other .NET Framework datatypes as values for OracleParameter without specifying either the DbType or the OracleDbType raises an exception because inferring DbType and OracleDbType from other .NET Framework datatypes is not supported.

Table 3–11 shows the inference of DbType and OracleDbType from Value when type of Value is one of Oracle.DataAccess.Types.

Table 3–11 Inference of DbType and OracleDbType from Value (ODP.NET Types)

Value (Oracle.DataAccess.Types)	System.Data.DbType	OracleDbType
OracleBFile	Object	BFile
OracleBinary	Binary	Raw
OracleBlob	Object	Blob
OracleClob	Object	Clob
OracleDate	Date	Date
OracleDecimal	Decimal	Decimal
OracleIntervalDS	Object	IntervalDS
OracleIntervalYM	Int64	IntervalYM
OracleRefCursor	Object	RefCursor
OracleString	String	Varchar2
OracleTimeStamp	DateTime	TimeStamp
OracleTimeStampLTZ	DateTime	TimeStampLTZ
OracleTimeStampTZ	DateTime	TimeStampTZ
OracleXmlType	String	XmlType

PL/SQL Associative Array

ODP.NET supports PL/SQL Associative Array (formerly known as PL/SQL Index-By Tables) binding.

An application can bind an OracleParameter, as a PL/SQL Associative Array, to a PL/SQL stored procedure. The following OracleParameter properties are used for this feature.

- CollectionType

  This property must be set to OracleCollectionType.PLSQLAssociativeArray to bind a PL/SQL Associative Array.

- ArrayBindSize

  This property is ignored for the fixed-length element types (such as Int32).

For variable-length element types (such as `Varchar2`), each element in the `ArrayBindSize` property specifies the size of the corresponding element in the `Value` property.

For `Output` parameters, `InputOutput` parameters, and return values, this property must be set for variable-length variables.

- `ArrayBindStatus`

  This property specifies the execution status of each element in the `OracleParameter.Value` property.

- `Size`

  This property specifies the maximum number of elements to be bound in the PL/SQL Associative Array.

- `Value`

  This property must either be set to an array of values or null or `DBNull.Value`.

Code Example

This example binds three `OracleParameter` objects as PL/SQL Associative Arrays: `Param1` as an `In` parameter, `Param2` as an `InputOutput` parameter, and `Param3` as an `Output` parameter.

PL/SQL Package : My Pack

```
CREATE PACKAGE MYPACK AS
TYPE AssocArrayVarchar2_t is table of VARCHAR(20) index by BINARY_INTEGER;
PROCEDURE TestVarchar2(
    Param1 IN     AssocArrayVarchar2_t,
    Param2 IN OUT AssocArrayVarchar2_t,
    Param3 OUT    AssocArrayVarchar2_t);
END MYPACK;
```

PL/SQL Package Body : My Pack

```
CREATE PACKAGE BODY MYPACK AS
PROCEDURE TestVarchar2(
    Param1 IN     AssocArrayVarchar2_t,
    Param2 IN OUT AssocArrayVarchar2_t,
    Param3 OUT    AssocArrayVarchar2_t)
    IS
    i integer;
    BEGIN
```

```
      -- copy a few elements from y to z
      Param3(1) := Param2(1);
      Param3(2) := NULL;
      Param3(3) := Param2(3);

-- copy all elements from x to y
      Param2(1) := Param1(1);
      Param2(2) := Param1(2);
      Param2(3) := Param1(3);

      FOR i IN 1..3 LOOP
        insert into T1 values(i, Param2(i));
      END LOOP;

      FOR i IN 1..3 LOOP
        select COL2 into Param2(i) from T2 where COL1 = i;
      END LOOP;
    END TestVarchar2;
END MYPACK;
```

ODP.NET **Example**

```
public void BindAssocArray()
{
...

      OracleCommand cmd = new OracleCommand(
        "begin MyPack.TestVarchar2(:1, :2, :3); end;", con);

      OracleParameter Param1 = cmd.Parameters.Add(...);
      OracleParameter Param2 = cmd.Parameters.Add(...);
      OracleParameter Param3 = cmd.Parameters.Add(...);

      Param1.Direction = ParameterDirection.Input;
      Param2.Direction = ParameterDirection.InputOutput;
      Param3.Direction = ParameterDirection.Output;

      // Specify that we are binding PL/SQL Associative Array
      Param1.CollectionType = OracleCollectionType.PLSQLAssociativeArray;
      Param2.CollectionType = OracleCollectionType.PLSQLAssociativeArray;
      Param3.CollectionType = OracleCollectionType.PLSQLAssociativeArray;

      // Setup the values for PL/SQL Associative Array
      Param1.Value = new string[3]{"First Element",
```

```
                                "Second Element ",
                                "Third Element ",
      Param2.Value = new string[3]{"First Element",
                                "Second Element ",
                                "Third Element ",
      Param3.Value = null;

      // Specify the maximum number of elements in the PL/SQL Associative Array
      Param1.Size = 3;
      Param2.Size = 3;
      Param3.Size = 3;

      // Setup the ArrayBindSize for Param1
      Param1.ArrayBindSize = new int[3]{13, 14, 13};

      // Setup the ArrayBindStatus for Param1
      Param1.ArrayBindStatus = new OracleParameterStatus[3]{
        OracleParameterStatus.Success,
        OracleParameterStatus.Success,
        OracleParameterStatus.Success};

      // Setup the ArrayBindSize for Param2
      Param2.ArrayBindSize = new int[3]{20, 20, 20};

      // Setup the ArrayBindSize for Param3
      Param3.ArrayBindSize = new int[3]{20, 20, 20};

      // execute the cmd
      cmd.ExecuteNonQuery();

      //print out the parameter's values
...
    }
```

Array Binding

The array bind feature enables applications to bind arrays of a type using the `OracleParameter` class. Using the array bind feature, an application can insert multiple rows into a table in a single database round-trip.

The following code example inserts three rows into the `Dept` table with a single database round-trip. The `OracleCommand ArrayBindCount` property defines the number of elements of the array to use when executing the statement.

```
// C#
...
// Create an array of values that need to be inserted
int[] myArrayDeptNo = new int[3]{10, 20, 30};

// Set the command text on an OracleCommand object
cmd.CommandText = "insert into dept(deptno) values (:deptno)";
// Set the ArrayBindCount to indicate the number of values
cmd.ArrayBindCount = 3;

// Create a parameter for the array operations
OracleParameter prm = new OracleParameter("deptno", OracleDbType.Int32);
prm.Direction = ParameterDirection.Input;
prm.Value     = myArrayDeptNo;

// Add the parameter to the parameter collection
cmd.Parameters.Add(prm);

// Execute the command
cmd.ExecuteNonQuery();
```

See Also: "Value" on page 4-276 for more information

OracleParameter Array Bind Properties The `OracleParameter` object provides two properties for granular control when using the array bind feature:

- `ArrayBindSize` Property

  The `ArrayBindSize` property is an array of integers specifying the maximum size for each corresponding value in an array. The `ArrayBindSize` property is similar to the `Size` property of an `OracleParameter` except `ArrayBindSize` specifies the size for each value in an array.

  Before the execution, the application must populate `ArrayBindSize`; after the execution, ODP.NET populates the `ArrayBindSize`.

  `ArrayBindSize` is used only for parameter types that have variable length such as `Clob`, `Blob` and `Varchar2`. The size is represented in bytes for binary datatypes and characters for the Unicode string types. The count for string types does not include the terminating character. The size is inferred from the actual size of the value, if it is not explicitly set. For an output parameter, the size of each value is set by ODP.NET. The `ArrayBindSize` property is ignored for fixed length datatypes.

- `ArrayBindStatus` Property

  The `ArrayBindStatus` property is an array of `OracleParameterStatus` values specifying status of each corresponding value in an array for a parameter. This property is similar to the `Status` property of `OracleParameter`, except that `ArrayBindStatus` specifies the status for each value in an array.

  Before the execution, the application must populate the `ArrayBindStatus` property and after the execution, ODP.NET populates it. Before the execution, an application using `ArrayBindStatus` can specify a `NULL` value for the corresponding element in the array for a parameter. After the execution, ODP.NET populates the `ArrayBindStatus` array, indicating whether the corresponding element in the array has a `NULL` value or if data truncation occurred when the value was fetched.

Error Handling for Array Binding If an error occurs during an Array Bind execution, it can be difficult to determine which element in the `Value` property caused the error. ODP.NET provides a way to determine the row where the error occurred, making it easier to find the element in the row that caused the error.

When an `OracleException` is thrown during an Array Bind execution, the `OracleErrorCollection` contains one or more `OracleError` objects. Each of these `OracleError` objects represents an individual error that occurred during the execution and contains a provider-specific property, `ArrayBindIndex`, which indicates the row number at which the error occurred.

Code Snippet

```
try {
  / An Array Bind execution errors out
}

catch (OracleException e)
{
  Console.WriteLine ("OracleException {0} occured", e.Message);
  for (int i = 0; i < e.Errors.Count; i++)
    Console.WriteLine("Array Bind Error {0} occured at Row Number {1}",
e.Errors[i].Message,
      e.Errors[i].ArrayBindIndex);
}
```

See Also: "ArrayBindIndex" on page 4-185 for more information

OracleParameterStatus Enumeration Types Table 3–12 provides different values for `OracleParameterStatus` enumeration.

Table 3–12 OracleParameterStatus Members

Member Names	Description
`Success`	For input parameters, it indicates that the input value has been assigned to the column. For output parameters, it indicates that the provider assigned an intact value to the parameter.
`NullFetched`	Indicates that a `NULL` value has been fetched from a column or an `OUT` parameter.
`NullInsert`	Indicates that a `NULL` value is to be inserted into a column.
`Truncation`	Indicates that truncation has occurred when fetching the data from the column.

PL/SQL REF CURSOR and OracleRefCursor

The `REF CURSOR` is a datatype in the Oracle PL/SQL language. It represents a cursor or a result set in the Oracle database. The `OracleRefCursor` is a corresponding ODP.NET type for the `REF CURSOR` type.

This section discusses the following aspects of using `REF CURSOR` and `OracleRefCursor` objects:

- Obtaining an OracleRefCursor
- Obtaining a REF CURSOR
- Populating an OracleDataReader from a REF CURSOR
- Populating the DataSet From a REF CURSOR
- Populating an OracleRefCursor From a REF CURSOR
- Updating a DataSet Obtained From a REF CURSOR
- Behavior of ExecuteScalar Method for REF CURSOR

Obtaining an OracleRefCursor

There are no constructors for `OracleRefCursor` objects. They can only be acquired as parameter values from PL/SQL stored procedures, stored functions, or anonymous blocks.

An `OracleRefCursor` is a connected object. The connection used to execute the command returning a `OracleRefCursor` object is required for its lifetime. Once the connection associated with an `OracleRefCursor` is closed, the `OracleRefCursor` cannot be used.

Obtaining a REF CURSOR

A `REF CURSOR` can be obtained as an `OracleDataReader`, `DataSet`, or `OracleRefCursor`. If the `REF CURSOR` is obtained as an `OracleRefCursor` object, it can be used to create an `OracleDataReader` or populate a `DataSet` from it. When accessing a `REF CURSOR`, always bind as a `OracleDbType.RefCursor`.

Populating an OracleDataReader from a REF CURSOR

An Oracle `REF CURSOR` can be obtained as an `OracleDataReader` by calling the `OracleCommand ExecuteReader` method. The output parameter with the `OracleDbType` property set is bound to `OracleDbType.RefCursor`. None of the output parameters of type `OracleDbType.RefCursor` are populated after the `ExecuteReader` is invoked.

If there are multiple output `REF CURSOR` parameters, use the `OracleDataReader NextResult` method to access the next `REF CURSOR`. The `OracleDataReader NextResult` method provides sequential access to the `REF CURSORs`; only one `REF CURSOR` can be accessed at a given time.

The order in which `OracleDataReader` objects are created for the corresponding `REF CURSOR` depends on the order in which the parameters are bound. If a PL/SQL stored function returns a `REF CURSOR`, then it becomes the first `OracleDataReader` and all the output `REF CURSOR` objects follow the order in which the parameters are bound.

Populating the DataSet From a REF CURSOR

For the `Fill` method to populate the `DataSet` properly, the `SelectCommand` of the `OracleDataAdapter` must be bound with an output parameter of type `OracleDbType.RefCursor`. If the `Fill` method is successful, the `DataSet` is populated with a `DataTable` that represents a `REF CURSOR`.

If the command execution returns multiple `REF CURSORs`, the `DataSet` is populated with multiple `DataTables`.

Populating an OracleRefCursor From a REF CURSOR

When `ExecuteNonQuery` is invoked on a command that returns one or more `REF CURSORs`, each of the `OracleCommand` parameters that are bound as `OracleDbType.RefCursor` gets a reference to an `OracleRefCursor` object.

To create an `OracleDataReader` from an `OracleRefCursor` object, invoke `GetDataReader` from an `OracleRefCursor` object. Subsequent calls to `GetDataReader` return the reference to the same `OracleDataReader`.

To populate a `DataSet` with an `OracleRefCursor` object, the application can invoke an `OracleDataAdapter Fill` method that takes an `OracleRefCursor` object.

When multiple `REF CURSORs` are returned from a command execution as `OracleRefCursor` objects, the application can choose to create an `OracleDataReader` or populate a `DataSet` with a particular `OracleRefCursor` object. All the `OracleDataReaders` or `DataSet` created from the `OracleRefCursor` are active at the same time and can be accessed in any order.

Updating a DataSet Obtained From a REF CURSOR

`REF CURSORs` are not updatable. However, data that is retrieved into a `DataSet` can be updated. Therefore, the `OracleDataAdapter` requires a custom SQL statement to flush any `REF CURSOR` data updates to the database.

The `OracleCommandBuilder` cannot be used to generate SQL for `REF CURSOR` updates.

Behavior of ExecuteScalar Method for REF CURSOR

`ExecuteScalar` returns the return value of a stored function or the first bind parameter of a stored procedure or an anonymous PL/SQL block. Therefore, if the `REF CURSOR` is not the return value of a stored function or the first bind parameter of a stored procedure or an anonymous PL/SQL block, the `REF CURSOR` is ignored by `ExecuteScalar`.

However, if the `REF CURSOR` is a return value of a stored function or the first bind parameter of a stored procedure or an anonymous PL/SQL block, the value of the first column of the first row in the `REF CURSOR` is returned.

> **See Also:** *Oracle Database Application Developer's Guide - Large Objects* for more information

LOB Support

ODP.NET provides an easy and optimal way to access and manipulate large datatypes. Oracle Database supports large character and large binary datatypes.

Large Character Datatypes

- `CLOB` - Character data can store up to 4 gigabytes (4 GB).
- `NCLOB` - Unicode National character set data can store up to 4 gigabytes.

Large Binary Datatypes

- `BLOB` - Unstructured binary data can store up to 4 gigabytes.
- `BFILE` - Binary data stored in external file can store up to 4 gigabytes.

> **Note:** `LONG` and `LONG RAW` datatypes are made available for backward compatibility in Oracle9*i*, but should not be used in new applications.

ODP.NET provides three objects for LOBs for manipulating LOB data: `OracleBFile`, `OracleBlob`, and `OracleClob`.

Table 3–13 shows the proper ODP.NET class to use for a particular Oracle LOB type.

Table 3–13 ODP.NET LOB Objects

Oracle LOB Type	ODP.NET LOB object
BFILE	OracleBFile object
BLOB	OracleBlob object
CLOB	OracleClob object
NCLOB	OracleClob object

The ODP.NET LOB objects can be obtained by calling the proper typed accessor on the OracleDataReader or as an output parameter on a command execution with the proper bind type.

All ODP.NET LOB objects inherit from the .NET Stream class to provide generic Stream operations. The LOB data (except for BFILEs) can be updated using the ODP.NET LOB objects by using methods such as Write. Data is not cached in the LOB objects when read and write operations are carried out. Therefore, each Read or Write request incurs a server round-trip. The OracleClob overloads the Read method, providing two ways to read data from a CLOB. The Read method that takes a byte[] as the buffer populates it with CLOB data as Unicode byte array. The Read method that takes a char[] as the buffer populates it with Unicode characters.

Extensions can also be found on the OracleBFile object. An OracleBFile object must be explicitly opened using the OpenFile method before any data can be read from it. To close a previously opened BFILE, use the CloseFile method.

Every ODP.NET LOB object is a connected object and requires a connection during its lifetime. If the connection associated with a LOB object is closed, then the LOB object is not usable and should be disposed.

If an ODP.NET LOB object is obtained from an OracleDataReader through a typed accessor, then its Connection property is set with a reference to the same OracleConnection object used by the OracleDataReader. If a LOB object is obtained as an output parameter, then its Connection property is set with a reference to the same OracleConnection property used by the OracleCommand. If a LOB object is obtained by invoking an ODP.NET LOB object constructor to create a temporary LOB, the Connection property is set with a reference to the OracleConnection object provided in the constructor.

The ODP.NET LOB object `Connection` property is read-only and cannot be changed during its lifetime. In addition, the ODP.NET LOB types object can only be used within the context of the same `OracleConnection` referenced by the ODP.NET LOB object. For example, the ODP.NET LOB object's `Connection` must reference the same connection as the `OracleCommand` if the ODP.NET LOB object is a parameter of the `OracleCommand`. If that is not the case, ODP.NET raises an exception when the command is executed.

> **See Also:** *Oracle Database Application Developer's Guide - Large Objects* for complete information about Oracle9*i* LOBs and how to use them

Updating LOBs Using a DataSet

`BFILE` and `BLOB` data are stored in the `DataSet` as `byte` arrays while `CLOB` and `NCLOB` data are stored as `string`s. In a similar manner to other types, an `OracleDataAdapter` object can be used to fill and update LOB data changes along with the use of the `OracleCommandBuilder` for auto-generating SQL.

Note that an Oracle LOB column can store up to 4 GB of data. When the LOB data is fetched into the `DataSet`, the actual amount of LOB data the `DataSet` can hold for a LOB column is limited to the maximum size of a .NET string type, which is 2 GB. Therefore, when fetching LOB data that is greater than 2 GB, ODP.NET LOB objects must be used to avoid any data loss.

Updating LOBs Using OracleCommand and OracleParameter

To update LOB columns, LOB data can be bound as a parameter for SQL statements, anonymous PL/SQL blocks, or stored procedures. The parameter value can be set as a NET Framework type, ODP.NET type, or as an ODP.NET LOB object type. For example, when inserting a .NET string data into a LOB column in a Oracle9*i* database, that parameter can be bound as `OracleDbType.Varchar2`. For a parameter whose value is set to an `OracleClob` object, the parameter should be bound as `OracleDbType.Clob`.

Updating LOBs Using ODP.NET LOB Objects

Oracle BFILEs are not updatable and hence OracleBFile objects do not allow updates to BFILE columns.

Two requirements must be met to update LOB data using ODP.NET LOB objects.

1. A transaction must be started before a LOB column is selected.

   The transaction must be started using the BeginTransaction method on the OracleCommand before the command execution so that the lock can be released when OracleTransaction Commit or Rollback is invoked.

2. The row in which the LOB column resides must be locked; on a row by row basis or as part of an entire result set.

   a. Locking the entire result

      Add the FOR UPDATE clause to the end of the SELECT statement. After execution of the command, the entire result set is locked.

   b. Locking the row - There are two options:

      – Invoke one of OracleDataReader's typed accessors (GetOracleClobForUpdate or GetOracleBlobForUpdate) on the OracleDataReader to obtain an ODP.NET LOB object while also locking the current row.

        This approach requires a primary key, unique column(s), or a ROWID in the result set because the OracleDataReader must uniquely identify the row to re-select it for locking.

      – Execute an INSERT or an UPDATE statement that returns a LOB in the RETURNING clause.

Temporary LOBs

Temporary LOBs can be instantiated for BLOBs, CLOBs, and NCLOBs. To instantiate an ODP.NET LOB object that represents a temporary LOB, the OracleClob or the OracleBlob constructor can be used.

Temporary ODP.NET LOB objects can be used for the following purposes:

- To initialize and populate a LOB column with empty or non-empty LOB data.
- To pass a LOB type as an input parameter to a SQL statement, anonymous PL/SQL blocks, or stored procedure.

- To act as the source or the destination of data transfer between two LOB objects as in the `CopyTo` operation.

> **Note:** Temporary LOBs are not transaction aware. Commits and rollbacks do not affect the data referenced by a temporary LOB.

Globalization Support

ODP.NET globalization support enables applications to manipulate culture-sensitive data appropriately. This feature ensures proper string format, date, time, monetary, numeric, sort order, and calendar conventions depending on the Oracle globalization settings.

See Also: "OracleGlobalization Class" on page 4-212

This section includes the following:

Globalization Settings

An `OracleGlobalization` object can be used to represent the following:

Client Globalization Settings

Client globalization settings are derived from the Oracle globalization setting (`NLS_LANG`) in the Windows registry of the local computer. The client globalization parameter settings are read-only and remain constant throughout the lifetime of the application. The client globalization settings can be obtained by calling the `OracleGlobalization.GetClientInfo()` static method.

The following example retrieves the client globalization setting:

```
// C#
...
// GetClientInfo() is a static method on OracleGlobalization class
OracleGlobalization ClientGlob = OracleGlobalization.GetClientInfo();
```

The properties of the `OracleGlobalization` object provide the Oracle globalization value settings.

Session Globalization Settings

Session globalization parameters are initially identical to client globalization settings. Unlike client settings, session globalization settings can be updated. However, they can only be obtained after establishing a connection against the database server. The session globalization settings can be obtained by calling `GetSessionInfo()` on the `OracleConnection`. Invoking this method returns an instance of an `OracleGlobalization` object whose properties represent the globalization settings of the session.

When the `OracleConnection` object establishes a connection, it implicitly opens a session whose globalization parameters are initialized with those values specified by the client computer's Oracle globalization (or National Language Setting (NLS)) registry settings. The session settings are updatable and can change during its lifetime.

The following example changes the date format setting on the session:

```
// C#
...
OracleConnection con = new OracleConnection("User Id=scott;Password=tiger;");
con.Open();
OracleGlobalization SessionGlob = con.GetSessionInfo();

// SetSessionInfo updates the Session with the new value
SessionGlob.DateFormat = "YYYY/MM/DD";
con.SetSessionInfo(SessionGlob);
...
```

Thread-Based Globalization Settings

Thread-based globalization parameter settings are specific to each thread. Initially, these settings are identical to the client globalization parameters, but they can be changed as specified by the application. When ODP.NET Types are converted to and from strings, the thread-based globalization parameters are used, if applicable.

Thread-based globalization parameter settings are obtained by invoking the `GetThreadInfo` static method of the `OracleGlobalization` object. The `SetThreadInfo` static method of the `OracleGlobalization` object can be called to set the thread's globalization settings.

ODP.NET classes and structures rely solely on the `OracleGlobalization` settings when manipulating culture-sensitive data. They do not use .NET thread culture information. If the application uses only .NET types, `OracleGlobalization` settings have no effect. However, when conversions are made between ODP.NET types and .NET types, `OracleGlobalization` settings are used where applicable.

> **Note:** Changes to `System.Threading.Thread.CurrentThread.CurrentCulture` do not impact the settings of the `OracleGlobalization` settings of the thread or the session and vice versa.

The following code snippet shows how the thread's globalization settings are used by the ODP.NET Types:

```
...
OracleGlobalization ThreadGlob = OracleGlobalization.GetThreadInfo();
// set and validate the format
ThreadGlob.DateFormat = "YYYY-MM-DD";

// set the thread with the new format
OracleGlobalization.SetThreadInfo(ThreadGlob);

// create a new instance of OracleDate
OracleDate date = new OracleDate("2002-01-01");
...
```

The `OracleGlobalization` object validates property changes made to it. If an invalid value is used to set a property, an exception is thrown. Note that changes made to the `Territory` and `Language` properties change other properties of the `OracleGlobalization` object implicitly.

See Also: *Oracle Database Globalization Support Guide* for more information on the properties affected by `Territory` and `Language` Globalization settings

Globalization-Sensitive Operations

This section lists ODP.NET types and operations that are dependent on or sensitive to globalization settings.

Operations Dependent on Client Computer's Globalization Settings

The `OracleString` structure depends on the client computer's `OracleGlobalization` settings. The local computer's client character set is used when it converts a Unicode string to a `byte[]` in the `GetNonUnicode` method and when it converts a `byte[]` of ANSI characters to Unicode in the `OracleString` constructor which accepts a `byte[]`.

Operations Dependent on Thread Globalization Settings

The thread globalization settings are used by ODP.NET types whenever they are converted to and from .NET string types, where applicable. In most cases, the `ToString` method, the `Parse` static method, constructors that accept .NET string data, and conversion operators to and from .NET strings use specific thread globalization settings depending on the ODP.NET type used.

For example, the `OracleDate` type uses the `DateFormat` property of the thread globalization settings when the `ToString` method is invoked on it. This returns a `DATE` as a string in the format specified by the thread's settings.

For more details, read the remarks in Chapter 5 for the ODP.NET type methods that convert between ODP.NET types and .NET string types, to identify which thread globalization settings are used for that particular method.

See Also: Chapter 5, "Oracle.DataAccess.Types Namespace (ODP.NET Types)"

Operations Sensitive to Session Globalization Parameters

Session globalization settings affect any data that is retrieved from or sent to the server as a string.

For example, if a `DATE` column is selected with the `TO_CHAR()` function applied on it, the `DATE` column data will be a string in the date format specified by the `DateFormat` of the session globalization settings. Transmitting data in the other

direction, the string data that is to be inserted into the DATE column, must be in the format specified by the DateFormat property of the session globalization settings.

The session globalization settings also affect data that is retrieved into the DataSet as a string using Safe Type Mapping. If the type is format-sensitive, the strings are always in the format specified by the session globalization settings.

For example, VARCHAR2 and CHAR data are not affected by session settings since no format is applicable for these types. However, the DateFormat and NumericCharacters properties can impact the string representation of DATE and NUMBER types, respectively, when they are retrieved as strings from the database server through safe type mapping.

See Also: "OracleDataAdapter Safe Type Mapping" on page 3-46

Guaranteeing Uniqueness in Updating DataSet to Database

This section describes how the OracleDataAdapter configures the PrimaryKey and Constraints properties of the DataTable which guarantee uniqueness when the OracleCommandBuilder is updating DataSet changes to the database.

Using the OracleCommandBuilder object to dynamically generate DML statements to be executed against the database is one of the ways to reconcile changes made in a single DataTable with the database.

In this process, the OracleCommandBuilder must not be allowed to generate DML statements that may affect (update or delete) more that a single row in the database when reconciling a single DataRow change. Otherwise the OracleCommandBuilder could corrupt data in the database.

To guarantee that each DataRow change affects only a single row, there must be a set of DataColumns in the DataTable for which all rows in the DataTable have a unique set of values. The set of DataColumns indicated by the properties DataTable.PrimaryKey and DataTable.Constraints meet this requirement. The OracleCommandBuilder determines uniqueness in the DataTable by checking whether the DataTable.PrimaryKey is non-null or if there exists a UniqueConstraint in the DataTable.Constraints collection.

This discussion first explains what constitutes uniqueness in DataRows and then explains how to maintain that uniqueness while updating, through DataTable property configuration.

This section includes the following topics:

What Constitutes Uniqueness in DataRows?

This section describes the minimal conditions that must be met to guarantee uniqueness of `DataRows`. The condition of uniqueness must be guaranteed before the `DataTable.PrimaryKey` and `DataTable.Constraints` properties can be configured, as described in the next section.

Uniqueness is guaranteed in a `DataTable` if any one of the following is true:

- All the columns of the primary key are in the select list of the `OracleDataAdapter.SelectCommand`.
- All the columns of a unique constraint are in the select list of the `OracleDataAdapter.SelectCommand`, with at least one involved column having a `NOT NULL` constraint defined on it.
- All the columns of a unique index are in the select list of the `OracleDataAdapter.SelectCommand`, with at least one of the involved columns having a `NOT NULL` constraint defined on it.
- A `ROWID` is present in the select list of the `OracleDataAdapter.SelectCommand`.

Note: A set of columns, on which a unique constraint has been defined or a unique index has been created, require at least one non-nullable column for following reason; if all the columns of the column set are nullable, then multiple rows could exist which have a `NULL` value for each column in the column set. This would violate the uniqueness condition that each row has a unique set of values for the column set.

Configuring PrimaryKey and Constraints Properties

If the minimal conditions described in "What Constitutes Uniqueness in DataRows?" on page 3-44 are met, then the `DataTable.PrimaryKey` or `DataTable.Constraints` properties can be set.

After these properties are set, the `OracleCommandBuilder` can determine uniqueness in the `DataTable` by checking the `DataTable.PrimaryKey` property or the presence of a `UniqueConstraint` in the `DataTable.Constraints` collection. Once uniqueness is determined, `OracleCommandBuilder` can safely generate DML statements to perform updates.

The `OracleDataAdapter.FillSchema` method attempts to set these properties according to this order of priority:

1. If the primary key is returned in the select list, it is set as the `DataTable.PrimaryKey`.
2. If a set of columns that meets the following criteria is returned in the select list, it is set as the `DataTable.PrimaryKey`.

   Criteria: The set of columns has a unique constraint defined on it or a unique index created on it, with each column having a `NOT NULL` constraint defined on it.
3. If a set of columns that meets the following criteria is returned in the select list, a `UniqueConstraint` is added to the `DataTable.Constraints` collection, but the `DataTable.PrimaryKey` is not set.

   Criteria: The set of columns has a unique constraint defined on it or a unique index created on it, with at least one column having a `NOT NULL` constraint defined on it.
4. If a `ROWID` is part of the select list, it is set as the `DataTable.PrimaryKey`.

Additionally, `OracleDataAdapter.FillSchema` exhibits the following behaviors:

- Setting `DataTable.PrimaryKey` implicitly creates a `UniqueConstraint`.
- If there are multiple occurrences of a column in the select list and the column is also part of the `DataTable.PrimaryKey` or `UniqueConstraint`, or both, each occurrence of the column is present as part of the `DataTable.PrimaryKey` or `UniqueConstraint`, or both.

Updating Without PrimaryKey and Constraints Configuration

If the `DataTable.PrimaryKey` or `Constraints` properties have not been configured, for example, if the application has not called `OracleDataAdapter.FillSchema`, the `OracleCommandBuilder` directly checks the select list of the `OracleDataAdapter.SelectCommand` to determine if it guarantees uniqueness in the `DataTable`. However this check results in a server round-trip to retrieve the metadata for the `SELECT` statement of the `OracleDataAdapter.SelectCommand`.

Note that `OracleCommandBuilder` cannot update a `DataTable` created from PL/SQL statements because they do not return any key information in their metadata.

OracleDataAdapter Safe Type Mapping

The ODP.NET `OracleDataAdapter` provides the Safe Type Mapping feature because the following Oracle datatypes can potentially lose data when converted to their closely related .NET type:

- `NUMBER`
- `DATE`
- `TimeStamp` (refers to all `TimeStamp` objects)
- `INTERVAL DAY TO SECOND`

When populating Oracle data containing any of these types into a .NET `DataSet` there is a possibility of data loss. The `OracleDataAdapter` Safe Type Mapping feature prevents data loss. By setting the `SafeMapping` property appropriately, these types can be safely represented in the `DataSet`, as either of the following:

- .NET `byte[]` in Oracle format
- .NET `String`

Potential Data Loss

The following sections provide more detail about the types and circumstances where data can be lost.

Oracle NUMBER Type to .NET Decimal Type

The Oracle datatype `NUMBER` can hold up to 38 precisions whereas .NET `Decimal` type can hold up to 28 precisions. If a `NUMBER` datatype that has more than 28 precisions is retrieved into .NET decimal type, it loses precision.

Table 3–14 lists the maximums and minimums for Oracle `NUMBER` and .NET `Decimal`.

Table 3–14 Oracle NUMBER to .NET Decimal Comparisons

	Oracle NUMBER	.NET Decimal
Maximum	9.99999999999999999999999999999999999 9999 e125	79,228,162,514,264,337,593,543,950,335
Minimum	-9.99999999999999999999999999999999999 99999 e125	-79,228,162,514,264,337,593,543,950,335

Oracle Date Type to .NET DateTime Type

The Oracle datatype `DATE` can represent dates in BC whereas .NET `DateTime` cannot. If a `DATE` that goes to BC get retrieved into .NET `DateTime` type, it loses data.

Table 3–15 lists the maximums and minimums for Oracle `Date` and .NET `DateTime`.

Table 3–15 Oracle Date to .NET DateTime Comparisons

	Oracle Date	.NET DateTime
Maximum	Dec 31, 9999 AD	Dec 31, 9999 AD 23:59:59.9999999
Minimum	Jan 1, 4712 BC	Jan 1, 0001 AD 00:00:00.0000000

Oracle TimeStamp Type to .NET DateTime Type

Similarly to `DATE`, Oracle datatype `TimeStamp` can represent a date in BC whereas .NET `DateTime` type cannot. If a `TimeStamp` that goes to BC is retrieved into .NET `DateTime` type, it loses data. Oracle `TimeStamp` type can represent values in units of e-9, whereas the .NET `DateTime` type can only represent values in units of e-7.

The Oracle TimeStamp with time zone datatype can store time zone information whereas .NET DateTime cannot.

Table 3–16 lists the maximums and minimums for Oracle TimeStamp and .NET DateTime.

Table 3–16 Oracle TimeStamp to .NET DateTime Comparisons

	Oracle TimeStamp	.NET DateTime
Maximum	Dec 31, 9999 AD 23:59:59.999999999	Dec 31, 9999 AD 23:59:59.9999999
Minimum	Jan 1, 4712 BC 00:00:00.000000000	Jan 1, 0001 AD 00:00:00.0000000

Oracle INTERVAL DAY TO SECOND to .NET TimeSpan

Similarly to DATE, the Oracle datatype INTERVAL DAY TO SECOND can represent dates in BC, whereas the .NET TimeSpan type cannot. If an INTERVAL DAY TO SECOND that goes to BC is retrieved into .NET TimeSpan type, it loses the data. The Oracle INTERVAL DAY TO SECOND type can represent values in units of e-9 whereas .NET TimeSpan type can only represent values in units of e-7.

Table 3–17 lists the maximums and minimums for Oracle INTERVAL DAY TO SECOND and .NET DateTime.

Table 3–17 Oracle INTERVAL DAY TO SECOND to .NET TimeSpan Comparisons

	Oracle INTERVAL DAY TO SECOND	.NET TmeSpan
Maximum	+999999999 23:59:59.999999999	+10675199 02:48:05.4775807
Minimum	-999999999 23:59:59.999999999	-10675199 02:48:05.4775808

SafeMapping Property

By default, Safe Type Mapping is disabled.

Using Safe Type Mapping

To use the Safe Type Mapping functionality, the OracleDataAdapter.SafeMapping property must be set with a hashtable of key-value pairs. The key-value pairs must map database table column names (of type string) to a .NET type (of type Type). ODP.NET supports safe type mapping to byte[] and String types. Any other type mapping causes an exception.

In situations where the column names are not known at design time, an asterisk ("*") can be used to map all occurrences of database types to a safe .NET type where it is needed. If both the valid column name and the asterisk are present, the column name is used.

Note:

- Database table column names are case sensitive.
- Column names in the hashtable that correspond to invalid column names are ignored.

Mapping to a .NET String

The safe type mapping as a string is more readable without further conversion. Converting certain Oracle datatypes to a string requires extra conversion, which can be slower than converting it to a `byte[]`. Conversion of .NET strings back to ODP.NET types relies on the formatting information of the session.

OracleDataAdapter Requery Property

The `OracleDataAdapter Requery` property controls whether queries are reexecuted for `OracleDataAdapter Fill` calls after the initial `Fill` call.

The `OracleDataAdapter Fill` method allows appending or refreshing data in the `DataSet`. When appending the `DataSet` using the same query with subsequent `Fill` calls, it may be desired not to reexecute the query.

When the `Requery` property is set to `true`, each subsequent `Fill` call reexecutes the query and fills the `DataSet`. It is an expensive operation and if the reexecution is not required, set `Requery` to `false`. If any of the `SelectCommand` properties or associated parameters needs to be changed, `Requery` must be `true`.

When the `Requery` property is set to `false`, the `DataSet` has the entire data as a snapshot at a particular time. The query is executed only for the first `Fill` call, subsequent `Fill` calls fetch the data from a cursor opened with the first execution of the query. This feature is only supported for forward-only fetches. `Fill` calls that try to fetch rows before the last fetched row raise an exception. The connection used for the first `Fill` call must be available for subsequent `Fill` calls.

When filling a `DataSet` with a `OracleRefCursor`, the `Requery` property can be used in a similar manner. When the `Requery` property is set to `false`, both the connection used for the first `Fill` and the `OracleRefCursor` must be available for the subsequent `Fill` calls.

See Also:

- "Requery" on page 4-97
- "SelectCommand" on page 4-100

Debug Tracing

ODP.NET provides debug tracing support, which allows logging of all the ODP.NET activities into a trace file. Different levels of tracing are available.

The provider can record the following information:

- Entry and Exit information for the ODP.NET public methods.
- User provided SQL statements as well as any SQL statements modified by the provider.
- Connection Pooling statistics such as Enlistment and Delistment.
- Thread ID (entry and exit).

Registry Settings for Tracing Calls

The following registry settings should be configured under

`HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOME`*`ID`*`\ODP.NET\HOME`

where *`ID`* is the appropriate Oracle Home.

TraceFileName

The valid values for `TraceFileName` are: 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 Also: "TraceOption" on page 3-51

TraceLevel

The valid values for `TraceLevel` are:

- `0` = None
- `1` = Entry, Exit, and SQL execution information
- `2` = Connection Pooling statistics
- `4` = Distributed Transactions (Enlistment and Delistment)

`TraceLevel` specifies the level of tracing in ODP.NET. Because tracing all the entry and exit calls for all the objects can be excessive, `TraceLevel` is provided to limit tracing to certain areas of the provider.

To obtain tracing on multiple objects, simply add the valid values. For example, if `TraceLevel` is set to `3`, trace information is logged for Entry, Exit, SQL, and Connection pooling information.

TraceOption

The valid values for `TraceOption` are:

- `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 the multiple trace files option is requested, a Thread ID is appended to the filename provided to create a trace file for each thread.

ODP.NET XML Support

From Oracle8*i* release 3 (8.1.7) and on, Oracle Database allows the extraction of data from relational and object-relational tables and views as XML documents. The use of XML documents for insert, update, and delete operations to the database server is also allowed.

With Oracle9*i* release 2 (9.2), Oracle Database supports XML natively in the database, through Oracle XML Database (Oracle XML DB), a distinct group of technologies related to high-performance XML storage and retrieval. Oracle XML DB is an evolution of the database that encompasses both SQL and XML data models in a highly interoperable manner, providing native XML support.

> **Note:** For database releases 8.1.7 and 9.0.1 only, certain `OracleCommand` methods require Oracle XML Developer's Kit (Oracle XDK) release 9.2 (Oracle XDK) or higher, to be installed in the database. The XDK can be downloaded from Oracle Technology Network (OTN).

For samples related to ODP.NET XML support, see the following directory:

*ORACLE_BASE**ORACLE_HOME*\ODP.NET\Samples

This section includes these topics:

- Supported XML Features
- OracleXmlType and Connection Dependency
- Updating XMLType Data in the Database Server
- Updating XML Data in OracleXmlType
- Special Characters in XML
- Retrieving Query Result Set as XML
- Data Manipulation Using XML

Supported XML Features

XML support in ODP.NET provides the following features:

- Store XML data natively in the database server as the Oracle database native type, `XMLType`.
- Access relational and object-relational data as XML data from an Oracle Database instance into Microsoft .NET environment, process the XML using Microsoft .NET framework.
- Save changes to the database server using XML data.

For the .NET application developer, these features include the following:

- Enhancements to the `OracleCommand`, `OracleConnection`, and `OracleDataReader` classes
- The following XML-specific classes:
  - `OracleXmlType` Class

`OracleXmlType` objects are used to retrieve Oracle native `XMLType` data.

- `OracleXmlStream` Class

  `OracleXmlStream` objects are used to retrieve XML data from `OracleXmlType` objects as a read-only .NET `Stream` object.

- `OracleXmlQueryProperties` Class

  `OracleXmlQueryProperties` objects represent the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Query`.

- `OracleXmlSaveProperties` Class

  `OracleXmlSaveProperties` objects represent the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Insert`, `Update`, or `Delete`.

See Also:

- "OracleCommand Class" on page 4-5
- "OracleXmlType Class" on page 5-646
- "OracleXmlStream Class" on page 5-630
- "OracleXmlQueryProperties Class" on page 4-336
- "OracleXmlSaveProperties Class" on page 4-345
- *Oracle XML DB Developer's Guide*

OracleXmlType and Connection Dependency

The read-only `Connection` property of the `OracleXmlType` object holds a reference to the `OracleConnection` object used to instantiate the `OracleXmlType` object.

How the `OracleXmlType` object obtains a reference to an `OracleConnection` object depends on how the `OracleXmlType` object is instantiated:

- Instantiated from an `OracleDataReader` using the `GetOracleXmlType`, `GetOracleValue`, or `GetOracleValues` method:

  The `Connection` property is set with a reference to the same `OracleConnection` object used by the `OracleDataReader`.

- Instantiated by invoking an `OracleXmlType` constructor with one of the parameters of type `OracleConnection`:

  The `Connection` property is set with a reference to the same `OracleConnection` object provided in the constructor.

- Instantiated by invoking an `OracleXmlType(OracleClob)` constructor:

  The `Connection` property is set with a reference to the `OracleConnection` object used by the `OracleClob` object.

An `OracleXmlType` object that is associated with one connection cannot be used with a different connection. For example, if an `OracleXmlType` object is obtained using `OracleConnection A`, that `OracleXmlType` object cannot be used as an input parameter of a command that uses `OracleConnection B`. By checking the `Connection` property of the `OracleXmlType` objects, the application can ensure that `OracleXmlType` objects are used only within the context of the `OracleConnection` referenced by its connection property. Otherwise, ODP.NET raises an exception.

Updating XMLType Data in the Database Server

Updating `XMLType` columns does not require a transaction. However, encapsulating the entire database update process within a transaction is highly recommended. This allows the updates to be rolled back if there are any errors.

`XMLType` columns in the database can be updated using the Oracle Data Provider for .NET in several ways:

- Updating with DataSet, OracleDataAdapter, and OracleCommandBuilder
- Updating with OracleCommand and OracleParameter

Updating with DataSet, OracleDataAdapter, and OracleCommandBuilder

If the `XMLType` column is fetched into the `DataSet`, the `XMLType` data is represented as a .NET `String`.

Modifying `XMLType` data in the `DataSet` does not require special treatment. `XMLType` data can be modified in the same way as any data that is stored in the `DataSet`. When a change is made and `OracleDataAdapter.Update()` is invoked, the `OracleDataAdapter` ensures that the `XMLType` data is handled properly. `OracleDataAdapter` uses any custom SQL `INSERT`, `UPDATE`, or `DELETE` statements that are provided. Otherwise, valid SQL statements are generated by the `OracleCommandBuilder` as needed to **flush** the changes to the database server.

Updating with OracleCommand and OracleParameter

OracleCommand provides a powerful way of updating XMLType data, especially with the use of OracleParameter. To update columns in a database table, the new value for the column can be passed as an input parameter of a command.

Input Binding To update an XMLType column in the database, a SQL statement can be executed using static values. In addition, input parameters can be bound to SQL statements, anonymous PL/SQL blocks, or stored procedures to update XMLType columns. The parameter value can be set as .NET Framework Types, ODP.NET Types, or OracleXmlType objects.

While XMLType columns can be updated using the OracleXmlType object, having an instance of an OracleXmlType object does not guarantee that the XMLType column in the database can be updated.

Setting XMLType Column to NULL Value Applications can set an XMLType column in the database to a NULL value, with or without input binding, as follows:

- Setting NULL values in an XMLType column with Input Binding

  To set the XMLType column to NULL, the application can bind an input parameter whose value is DBNull.Value. This indicates to the OracleCommand that a NULL value is to be inserted.

  Passing in a null OracleXmlType object as an input parameter does not insert a NULL into the XMLType column. In this case, the OracleCommand raises an exception.

- Setting NULL Values in an XMLType Column without Input Binding

  The following example demonstrates setting NULL values in an XMLType column without input binding:

  ```
  // Create a table with an XMLType column in the database
  CREATE TABLE XMLTABLE(NUM_COL number, XMLTYPE_COL xmltype);
  ```

  An application can set a NULL value in the XMLType column by explicitly inserting a NULL or by not inserting anything into that column as in the following examples:

  ```
  insert into xml_table(xmltype_col) values(NULL);
  update xml_table t set t.xmltype_col=NULL;
  ```

Setting XMLType Column to Empty XML Data The `XMLType` column can be initialized with empty XML data, using a SQL statement:

```
// Create a table with an XMLType column in the database
CREATE TABLE XMLTABLE(NUM_COL number, XMLTYPE_COL xmltype);

INSERT INTO XML_TABLE (NUM_COL, XMLTYPE_COL) VALUES (4,
      XMLType.createxml('<DOC/>'));
```

Updating XML Data in OracleXmlType

There are several ways that XML data can be updated in an `OracleXmlType` object.

- The XML data can be updated by passing an XPATH expression and the new value to the `Update` method on the `OracleXmlType` object.
- The XML data can be retrieved on the client side as the .NET Framework `XmlDocument` object using the `GetXmlDocument` method on the `OracleXmlType` object. This XML data can then be manipulated using suitable .NET Framework classes. A new `OracleXmlType` can be created with the updated XML data from the .NET Framework classes. This new `OracleXmlType` is bound as an input parameter to an update or insert statement.

Special Characters in XML

The following characters have special meaning in XML. For more information, refer to the XML 1.0 specifications

Table 3–18 Special Characters

Special Character	Meaning in XML	Entity Encoding
<	Begins an XML tag	&lt;
>	Ends an XML tag	&gt;
"	Quotation mark	&quot;
'	Apostrophe or single quotation mark	&apos;
&	Ampersand	&amp;

When these characters appear as data in an XML element, they are replaced with their equivalent entity encoding.

Also certain characters are not valid in XML element names. When SQL identifiers (such as column names) are mapped to XML element names, these characters are converted to a sequence of hexadecimal digits, derived from the Unicode encoding of the character, bracketed by an introductory underscore, a lowercase **x** and a trailing underscore. For example, the space is not a valid character in an XML element name. If a SQL identifier contains a space character, then in the corresponding XML element name, the space character is replaced by `_x0020_`, which is based on Unicode encoding of the space character.

Retrieving Query Result Set as XML

This section discusses retrieving the result set from a SQL query as XML data.

Handling Date and Time Format

Table 3–19 lists the date and time format handling for different database releases.

Table 3–19 Database Release Date and Time Differences When Retrieving Data

Database Release	Date and Time Format Supported
Oracle8*i* release 3 (8.1.7) and Oracle9*i* release 1 (9.0.*x*)	Oracle `DATE`, `TIMESTAMP`, `TIMESTAMP WITH TIME ZONE` type data is always retrieved in the result XML document as the ISO Date and Time Format: `YYYY-MM-DDThh:mm:ss.sss` (ISO Format notation). The following string is the ISO Date and Time Format notation represented in the Oracle Date and Time Format notation: `YYYY-MM-DD"T"HH:MI:SS.FF3`.

Table 3–19 Database Release Date and Time Differences When Retrieving Data

Database Release	Date and Time Format Supported
Oracle9*i* release 2 (9.2.*x*)	Oracle `DATE` type data is retrieved in the format specified using the `NLS_DATE_FORMAT` in the session. `TIMESTAMP` and `TIMESTAMP WITH TIME ZONE` type data is retrieved in the format specified using the `NLS_TIMESTAMP_FORMAT` and the `NLS_TIMESTAMP_TZ_FORMAT` in the session. If the result XML document is used to save changes to database releases 8.1.7, 9.0.*x*, or 9.2.*x*, then all `DATE` and `TIMESTAMP` data must be retrieved in the XML document as the following ISO Date and Time Format: `YYYY-MM-DDThh:mm:ss.sss` (ISO Format notation). To do this, before the query is executed, the application must explicitly perform an `ALTER SESSION` command on the session for the following NLS session parameters: ■ `NLS_DATE_FORMAT` - Must be set to the following Oracle Date and Time Format: `YYYY-MM-DD"T"HH:MI:SS` ■ `NLS_TIMESTAMP_FORMAT` - Must be set to the following Oracle Date and Time Format: `YYYY-MM-DD"T"HH:MI:SS.FF3` ■ `NLS_TIMESTAMP_TZ_FORMAT` - Must be set to the following Oracle Date and Time Format: `YYYY-MM-DD"T"HH:MI:SS.FF3`
Oracle Database 10*g*	Oracle `DATE` type data is retrieved in the format specified using the `NLS_DATE_FORMAT` in the session. `TIMESTAMP` and `TIMESTAMP WITH TIME ZONE` data is retrieved in the format specified using the `NLS_TIMESTAMP_FORMAT` and the `NLS_TIMESTAMP_TZ_FORMAT` in the session. For interoperability with Oracle9*i* release 1 (9.0.*x*), Oracle9*i* release 2 (9.2.*x*), and Oracle 8.1.7, Oracle recommends that all `DATE` and `TIMESTAMP` data be retrieved in the XML document as the following ISO Date and Time Format: `YYYY-MM-DDThh:mm:ss.sss` (ISO Format notation).

Special Characters in Column Data

If any of the data in the select list columns in the query contain any characters with special meaning in XML (see Table 3–18), these characters are replaced with their corresponding entity encoding in the result XML document.

The following examples demonstrate how ODP.NET handles the angle bracket special characters in the column data:

```
// Create the following table
create table specialchars ("id" number, name varchar2(255));
```

```
insert into specialchars values (1, '<Jones>');

// Create the connection
string constr = "User Id=hr;Password=hr;Data Source=orcl";
OracleConnection conn = new OracleConnection(constr);
conn.Open();

// Create the command
OracleCommand cmd = new OracleCommand("", conn);

// Set the XML command type to query.
cmd.XmlCommandType = OracleXmlCommandType.Query;

// Set the SQL query
cmd.CommandText = "select * from specialchars";

// Set command properties that affect XML query behavior.
cmd.BindByName = true;

// Set the XML query properties
cmd.XmlQueryProperties.MaxRows = -1;

// Get the XML document as an XmlReader.
XmlReader xmlReader = cmd.ExecuteXmlReader();
XmlDocument xmlDocument = new XmlDocument();
xmlDocument.PreserveWhitespace = true;
xmlDocument.Load(xmlReader);
Console.WriteLine(xmlDocument.OuterXml);
```

The following XML document is generated for that table:

```
<?xml version = '1.0'?>
<ROWSET>
   <ROW>
      <id>1</id >
      <NAME>&lt;Jones&gt;</NAME>
   </ROW>
</ROWSET>
```

Special Characters In Table or View Name

If a table or view name has any non-alphanumeric characters other than an underscore (_), the table or view name must be enclosed in quotation marks.

For example, to select all entries from a table with the name `test'ing`, the `CommandText` property of the `OracleCommand` object must be set to the following string.

```
"select * from \"test'ing\"";
```

Case-Sensitivity in Column Name to XML Element Name Mapping

The mapping of SQL identifiers (column names) to XML element names is case sensitive and the element names are in exactly the same case as the column names of the table or view.

However, the root tag and row tag names are case insensitive. The following example demonstrates case-sensitivity in this situation:

```
//Create the following table
create table casesensitive_table ("Id" number, NAME varchar2(255));

//insert name and id
insert into casesensitive_table values(1, 'Smith');

The following XML document is generated:
<?xml version = '1.0'?>
  <ROWSET>
    <ROW>
      <Id>1</Id>
      <NAME>Smith</NAME>
    </ROW>
  </ROWSET>
```

Note that the element name for the **Id** column matches the case with the column name.

Column Name to XML Element Name Mapping

For each row generated by the SQL query, the SQL identifier (column name) maps to an XML element in the generated XML document.

The following example demonstrates this:

```
// Create the following table
create table emp_table (EMPLOYEE_ID NUMBER(4), LAST_NAME varchar2(25));
// Insert some data
insert into emp_table values(205, 'Higgins');
```

The SQL query, select * from `emp_table`, generates the following XML document:

```
<?XML version="1.0"?>
  <ROWSET>
    <ROW>
      <EMPLOYEE_ID>205</EMPLOYEE_ID>
      <LAST_NAME>Higgins</LAST_NAME>
    </ROW>
  </ROWSET>
```

The EMPLOYEE_ID and LAST_NAME database columns of the employees table map to the EMPLOYEE_ID and LAST_NAME elements of the generated XML document.

Retrieving Results from Oracle 8.1.7 When retrieving the query results as XML from an Oracle 8.1.7 database, the SQL identifiers in the query select-list cannot contain characters that are not valid in XML element names. To handle the lack of support for this feature in Oracle 8.1.7, the SQL query in the following example can be used to get a result as a XML document from the specialchars table:

```
select "some id" as "some_x0020_id", name from specialchars;
```

Retrieving Results from Oracle9*i* or Higher When retrieving the query results as XML from Oracle9*i* and higher, the SQL identifiers in the query select-list can contain characters that are not valid in XML element names. When these SQL identifiers (such as column names) are mapped to XML element names, each of these characters are converted to a sequence of hexadecimal digits, derived from the Unicode encoding of the characters, bracketed by an introductory underscore, a lower case x, and a trailing underscore.

Thus, with an Oracle9*i* database, the SQL query in the following example can be used to get a result as an XML document from the specialchars table:

```
select "some id", name from specialchars;
```

See Also: "Special Characters in XML" on page 3-56

Improving Default Mapping If this default mapping of SQL identifiers to XML element names is not adequate, you can improve the mapping by the following techniques:

- Modify the source. Create an object-relational view over the source schema, and make that view the new source.
- Use cursor subqueries and cast-multiset constructs in the SQL query.

- Create an alias for the column or attribute names in the SQL query. Prepend the aliases with an at sign (@) to map them to XML attributes instead of XML elements.
- Modify the XML Document. Use XSLT to transform the XML document. Specify the XSL document and parameters. The transformation is done automatically after the XML document is generated from the relational data. Note that this is not the best solution in terms of performance.
- Specify the name of the root tag and row tag used in the XML document.

Object-Relational Data

ODP.NET can generate an XML document for data stored in object-relational columns, tables, and views.

The following example demonstrates this:

```
// Create the following tables and types
CREATE TYPE "EmployeeType" AS OBJECT (EMPNO NUMBER, ENAME VARCHAR2(20));
/
CREATE TYPE EmployeeListType AS TABLE OF "EmployeeType";
/
CREATE TABLE mydept (DEPTNO NUMBER, DEPTNAME VARCHAR2(20),
              EMPLIST EmployeeListType)
              NESTED TABLE EMPLIST STORE AS EMPLIST_TABLE;
INSERT INTO mydept VALUES (1, 'depta',
              EmployeeListType("EmployeeType"(1, 'empa')));
```

The following XML document is generated for the table:

```
<?xml version = "1.0"?>
<ROWSET>
   <ROW>
      <DEPTNO>1</DEPTNO>
      <DEPTNAME>depta</DEPTNAME>
      <EMPLIST>
         <EmployeeType>
            <EMPNO>1</EMPNO>
            <ENAME>empa</ENAME>
         </EmployeeType>
      </EMPLIST>
   </ROW>
</ROWSET>
```

ODP.NET encloses each item in a collection element, with the database type name of the element in the collection. The `mydept` table has a collection in the `EMPLIST` database column and each item in the collection is of type `EmployeeType`. Therefore, in the XML document, each item in the collection is enclosed in the type name `EmployeeType`.

NULL values

If any database row has a column with a `NULL` value, then that column does not appear for that row in the generated XML document.

Data Manipulation Using XML

This section discusses making changes to the database using XML.

Handling of Date and Time Format

Table 3–20 lists the date and time format handling for different database releases.

Table 3–20 Database Release Date and Time Differences When Saving Data

Database Release	Date and Time Format Supported
Oracle8*i* release (8.1.7), Oracle9*i* release 1 (9.0.*x*), and Oracle9*i* release 2 (9.2.*x*)	All `DATE`, `TIMESTAMP` and `TIMESTAMP WITH TIME ZONE` type data must be specified in the XML document in the ISO Date and Time Format `YYYY-MM-DDThh:mm:ss.sss` (ISO Format notation).
Oracle Database 10*g*	All `DATE`, `TIMESTAMP` and `TIMESTAMP WITH TIME ZONE` type data must be specified in the XML document in the `NLS_DATE_FORMAT`, `NLS_TIMESTAMP_FORMAT`, and `NLS_TIMESTAMP_TZ_FORMAT` of the session, respectively. If the `DATE`, `TIMESTAMP`, and `TIMESTAMP WITH TIME ZONE` type data in the XML document is different from the session settings, then before saving the changes using the XML document, the application must explicitly perform an `ALTER SESSION` command on the session for the following NLS session parameters to reflect the format in the XML document: ▪ `NLS_DATE_FORMAT` ▪ `NLS_TIMESTAMP_FORMAT` ▪ `NLS_TIMESTAMP_TZ_FORMAT`

Saving Changes Using XML

Changes can be saved to database tables and views using XML data. However, insert, update, and delete operations cannot be combined in a single XML document. ODP.NET cannot accept a single XML document and determine which changes are inserts, updates, or deletes.

The inserts must be in an XML document containing only rows to be inserted, the updates only with rows to be updated, and the deletes only with rows to be deleted.

For example, using the `employees` table that comes with the HR sample schema, you can specify the following query:

```
select employee_id, last_name from employees where employee_id = 205;
```

The following XML document is generated:

```
<?xml version = '1.0'?>
<ROWSET>
   <ROW>
      <EMPLOYEE_ID>205</EMPLOYEE_ID>
      <LAST_NAME>Higgins</LAST_NAME>
   </ROW>
</ROWSET>
```

To change the name of employee `205` from **`Higgins`** to **`Smith`**, specify the `employees` table and the XML data containing the changes as follows:

```
<?xml version = '1.0'?>
<ROWSET>
   <ROW>
      <EMPLOYEE_ID>205</EMPLOYEE_ID>
      <LAST_NAME>Smith</LAST_NAME>
   </ROW>
</ROWSET>
```

Special Characters in Column Data

If the data in any of the elements in the XML document contains characters that have a special meaning in XML (see Table 3–18), these characters must be entity-encoded or escaped in the XML document, so that the data is stored correctly in the database table column. Otherwise, ODP.NET throws an exception.

The following examples demonstrate how ODP.NET handles the angle bracket special characters in the column data.

```
// Create the following table
create table specialchars ("id" number, name varchar2(255));
```

The following XML document can be used to insert values `(1, '<Jones>')` to the `specialchars` table:

```
<?xml version = '1.0'?>
 <ROWSET>
  <ROW>
   <id>1</id >
   <NAME>&lt;Jones&gt;</NAME>
  </ROW>
 </ROWSET>
```

Special Characters in Table or View Name

If a table or view name has any non-alphanumeric characters other than an underscore (_), the table or view name must be enclosed in quotation marks.

For example, to save changes to a table with the name `test'ing`, the `OracleCommand.XmlSaveProperties.TableName` property must be set to `"\"test'ing\""`.

Case-Sensitivity in XML Element Name to Column Name Mapping

For each XML element representing a row of data in the XML document, the child XML elements map to database column names. The mapping of the child element name to the column name is always case sensitive, but the root tag and row tag names are case insensitive. The following example demonstrates this case-sensitivity:

```
//Create the following table
create table casesensitive_table ("Id" number, NAME varchar2(255));
```

The following XML document can be used to insert values (`1`, `Smith`) into the `casesensitive_table`:

```
<?xml version = '1.0'?>
  <ROWSET>
    <ROW>
      <Id>1</Id>
      <NAME>Smith</NAME>
    </ROW>
```

```
  </ROWSET>
```

Note the element name for the **Id** column matches the case with the column name.

XML Element Name to Column Name Mapping

Oracle9*i* and higher handles the mapping of XML element names to column names differently from Oracle 8.1.7 when using XML for data manipulation in the database. This section demonstrate these differences with changes to the following `specialchars` table involving the some id column.

```
// Create the specialchars table
create table specialchars ("some id" number, name varchar2(255));
```

Note that the `specialchars` table has a some id column that contains a space character. The space character is not allowed in an XML element name.

Saving changes to Oracle 8.1.7 In this scenario, with an Oracle 8.1.7 database, in order to save changes to the `specialchars` table using an XML document, a view must be created over the table and the changes saved to the view using XML data.

The column names in the view corresponding to the some id column in the table can be either a column name with no invalid characters or the escaped column name as in the following example.

```
// Create the view with the escaped column name
create view view1(some_x0020_id, name) as select * from specialchars;

// Create the view with the column name with no invalid character
create view view2(someid, name) as select * from specialchars;
```

The following XML document can be used to insert values (`1`, `<Jones>`) into the `specialchars` table using `view1`:

```
<?xml version = '1.0'?>
  <ROWSET>
    <ROW>
      <SOME_X0020_id>1</SOME_X0020_id >
      <NAME>&lt;Jones&gt;</NAME>
    </ROW>
  </ROWSET>
```

The following XML document can be used to insert values (1, <Jones>) into the specialchars table using view2:

```
<?xml version = '1.0'?>
  <ROWSET>
    <ROW>
      <SOMEID>2</SOMEID>
      <NAME>&lt;Jones&gt;</NAME>
      </ROW>
  </ROWSET>
```

Saving Changes to Oracle9*i* or higher When an XML document is used to save changes to a table or view, the OracleCommand.XmlSaveProperties.UpdateColumnsList is used to specify the list of columns to update or insert.

With Oracle9*i* or higher, when an XML document is used to save changes to a column in a table or view and the corresponding column name contains any of the characters which are not valid in an XML element name, the escaped column name needs to be specified in the UpdateColumnsList property as in the following example.

The following XML document can be used to insert values (2, <Jones>) into the specialchars table.

```
<?xml version = '1.0'?>
  <ROWSET>
    <ROW>
      <some_x0020_id>2</some_x0020_id>
      <NAME>&lt;Jones&gt;</NAME>
    </ROW>
  </ROWSET>
```

The following code example specifies the list of columns to update or insert.

```
CmdObj.XmlCommandType = OracleXmlCommandType.Insert;
CmdObj.XmlSaveProperties.Table = "specialchars";
string[] ucols = new string[2];
ucols[0] = "some_x0020_id";
ucols[1] = "NAME";
CmdObj.XmlSaveProperties.UpdateColumnsList = ucols;
CmdObj.ExecuteNonQuery();
```

Improving Default Mapping If the default mapping is not adequate, you can improve the mapping by the following techniques:

- Modify the target. Create an object-relational view over the target schema, and make the view the new target.
- Modify the XML Document. Use XSLT to transform the XML document. Specify the XSL document and parameters. The transformation is done before the changes are saved. Note that this is not the best solution in terms of performance.
- Specify the name of the row tag used in the XML document.

Object-Relational Data

Changes in an XML document can also be saved to object-relational data. Each item in a collection can be specified in one of the following ways in the XML document:

- By enclosing the database type name of the item as the XML element name.
- By enclosing the name of the database column holding the collection with `_ITEM` appended as the XML element name.

Multiple Tables

Oracle Database does not support saving changes to multiple relational tables that have been joined together. In this case, Oracle recommends that you create a view on those relational tables, and then update that view. If the view is not updatable, triggers can be used instead.

> **See Also:** *Oracle Database SQL Reference* for the description and syntax of the `CREATE VIEW` command

Commits

When the changes in an XML document are made, either all the changes are committed, or if an error occurs, any changes that were made are rolled back.

4

Oracle.DataAccess.Client Namespace

This chapter describes the Oracle Data Provider classes.

This chapter contains these topics:

Overview of Oracle Data Provider Classes

Oracle Data Provider for .NET classes expose inherited, provider-specific, interface implementations of methods and properties.

ODP.NET provider-specific and interface implementations of methods and properties are described in detail. Inherited methods and properties are not described in detail unless they are overridden. See the Microsoft .NET Framework Class Library for detailed descriptions of inherited methods and properties.

Assembly and Namespace

Oracle Data Provider objects are provided in the `Oracle.DataAccess.Client` namespace of the `Oracle.DataAccess.dll` assembly.

Class Inheritance

Information on class inheritance is provided for each class. The following is an example of the inheritance summary for the `OracleConnection` class. It shows that the `OracleConnection` class inherits from the `Component` class, the `Component` class inherits from the `MarshalByRefObject` class, and the `MarshalByRefObject` class inherits from the `Object` class.

```
Object

  MarshalByRefObject

    Component

      OracleConnection
```

Interface Inheritance

Information on interface inheritance is provided in the class declaration. The following example of the `OracleConnection` declaration shows that it inherits from the `IDbConnection` and `ICloneable` interfaces.

Note that the declaration also indicates the class it derives from, which in this case is the `Component` class.

```
public sealed class OracleConnection : Component, IDbConnection, ICloneable
```

Syntax Used

The class descriptions in this guide use the C# syntax and datatypes. Check the related Visual Studio .NET Framework documentation for information on other .NET language syntax.

Oracle Data Provider Classes

This chapter describes the classes and public methods Oracle Data Provider for .NET exposes for ADO.NET programmers. They are:

OracleCommand Class

An `OracleCommand` object represents a SQL command, a stored procedure, or a table name. The `OracleCommand` object is responsible for formulating the request and passing it to the database. If results are returned, `OracleCommand` is responsible for returning results as an `OracleDataReader`, a .NET `XmlReader`, a .NET `Stream`, a scalar value, or as output parameters.

Class Inheritance

```
Object
  MarshalByRefObject
    Component
      OracleCommand
```

Declaration

```
// C#
public sealed class OracleCommand : Component, IDbCommand, ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The execution of any transaction-related statements from an `OracleCommand` is not recommended because it is not reflected in the state of the `OracleTransaction` object represents the current local transaction, if one exists.

`ExecuteXmlReader`, `ExecuteStream`, and `ExecuteToStream` methods are only supported for XML operations.

`ExecuteReader` and `ExecuteScalar` methods are not supported for XML operations.

Example

```
// C#
...
string conStr = "User Id=scott;Password=tiger;Data Source=oracle";

// Create the OracleConnection
```

```
OracleConnection con = new OracleConnection(conStr);
con.Open();

string cmdQuery = "select ename, empno from emp";

// Create the OracleCommand
OracleCommand cmd = new OracleCommand(cmdQuery);
cmd.Connection = con;
cmd.CommandType = CommandType.Text;

// Execute command, create OracleDataReader object
OracleDataReader reader = cmd.ExecuteReader();

while (reader.Read())
{
    // output Employee Name and Number
    Console.WriteLine("Employee Name : " + reader.GetString(0) + " , " +
                      "Employee Number : " + reader.GetDecimal(1));
}

// Dispose OracleDataReader object
reader.Dispose();

// Dispose OracleCommand object
cmd.Dispose();

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
...
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Members
- OracleCommand Constructors
- OracleCommand Static Methods
- OracleCommand Properties
- OracleCommand Public Methods

OracleCommand Members

`OracleCommand` members are listed in the following tables:

OracleCommand Constructors

`OracleCommand` constructors are listed in Table 4–1.

Table 4–1 OracleCommand Constructors

Constructor	Description
OracleCommand Constructors	Instantiates a new instance of `OracleCommand` class (Overloaded)

OracleCommand Static Methods

`OracleCommand` static methods are listed in Table 4–2.

Table 4–2 OracleCommand Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleCommand Properties

`OracleCommand` properties are listed in Table 4–3.

Table 4–3 OracleCommand Properties

Name	Description
AddRowid	Adds the `ROWID` as part of the select list

Table 4–3 OracleCommand Properties (Cont.)

Name	Description
ArrayBindCount	Specifies if the array binding feature is to be used and also specifies the maximum number of array elements to be bound in the `Value` property
BindByName	Specifies the binding method in the collection
CommandText	Specifies the SQL statement or stored procedure to run against the Oracle database or the XML data used to store changes to the Oracle database
`CommandTimeout`	*Not supported*
CommandType	Specifies the command type that indicates how the `CommandText` property is to be interpreted
Connection	Specifies the `OracleConnection` object that is used to identify the connection to execute a command
`Container`	Inherited from `Component`
FetchSize	Specifies the size of `OracleDataReader`'s internal cache to store result set data
InitialLOBFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns
Parameters	Specifies the parameters for the SQL statement or stored procedure
RowSize	Specifies the amount of memory needed by the `OracleDataReader` internal cache to store one row of data
XmlCommandType	Specifies the type of XML operation on the `OracleCommand`
XmlQueryProperties	Specifies the properties that are used when an XML document is created from the result set of a SQL query statement
XmlSaveProperties	Specifies the properties that are used when an XML document is used to save changes to the database

OracleCommand Public Methods

`OracleCommand` public methods are listed in Table 4–4.

Table 4–4 OracleCommand Public Methods

Public Method	Description
`Cancel`	*Not Supported*
Clone	Creates a copy of `OracleCommand` object
`CreateObjRef`	Inherited from `MarshalByRefObject`
CreateParameter	Creates a new instance of `OracleParameter` class
`Dispose`	Inherited from `Component`
`Equals`	Inherited from `Object` (Overloaded)
ExecuteNonQuery	Executes a SQL statement or a command using the `XmlCommandType` and `CommandText` properties and returns the number of rows affected
ExecuteReader	Executes a command (Overloaded)
ExecuteScalar	Returns the first column of the first row in the result set returned by the query
ExecuteStream	Executes a command using the `XmlCommandType` and `CommandText` properties and returns the results in a new `Stream` object
ExecuteToStream	Executes a command using the `XmlCommandType` and `CommandText` properties and appends the results as an XML document to the existing `Stream`
ExecuteXmlReader	Executes a command using the `XmlCommandType` and `CommandText` properties and returns the result as an XML document in a .NET `XmlTextReader` object
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
`Prepare`	*This method is a no-op*
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class

OracleCommand Constructors

`OracleCommand` constructors instantiate new instances of `OracleCommand` class.

Overload List:

- OracleCommand()

  This constructor instantiates a new instance of `OracleCommand` class.

- OracleCommand(string)

  This constructor instantiates a new instance of `OracleCommand` class using the supplied SQL command or stored procedure, and connection to the Oracle database.

- OracleCommand(string, OracleConnection)

  This constructor instantiates a new instance of `OracleCommand` class using the supplied SQL command or stored procedure, and connection to the Oracle database.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

OracleCommand()

This constructor instantiates a new instance of `OracleCommand` class.

Declaration

```
// C#
public OracleCommand();
```

Remarks

Default constructor.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

OracleCommand(string)

This constructor instantiates a new instance of `OracleCommand` class using the supplied SQL command or stored procedure, and connection to the Oracle database.

Declaration

```
// C#
public OracleCommand(string cmdText);
```

Parameters

- *cmdText*

  The SQL command or stored procedure to be executed.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

OracleCommand(string, OracleConnection)

This constructor instantiates a new instance of `OracleCommand` class using the supplied SQL command or stored procedure, and connection to the Oracle database.

Declaration

```
// C#
public OracleCommand(string cmdText, OracleConnection OracleConnection);
```

Parameters

- *cmdText*

  Specifies the SQL command or stored procedure to be executed.

- *OracleConnection*

  Specifies the connection to the Oracle database.

  See Also:

  - Oracle.DataAccess.Client Namespace
  - OracleCommand Class
  - OracleCommand Members

OracleCommand Static Methods

OracleCommand static methods are listed in Table 4–5.

Table 4–5 OracleCommand Static Methods

Methods	Description
Equals	Inherited from Object (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

OracleCommand Properties

OracleCommand properties are listed in Table 4–6.

Table 4–6 OracleCommand Properties

Name	Description
AddRowid	Adds the ROWID as part of the select list
ArrayBindCount	Specifies if the array binding feature is to be used and also specifies the maximum number of array elements to be bound in the Value property
BindByName	Specifies the binding method in the collection
CommandText	Specifies the SQL statement or stored procedure to run against the Oracle database or the XML data used to store changes to the Oracle database

Table 4–6 OracleCommand Properties (Cont.)

Name	Description
CommandTimeout	*Not supported*
CommandType	Specifies the command type that indicates how the `CommandText` property is to be interpreted
Connection	Specifies the `OracleConnection` object that is used to identify the connection to execute a command
Container	Inherited from `Component`
FetchSize	Specifies the size of `OracleDataReader`'s internal cache to store result set data
InitialLOBFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns
Parameters	Specifies the parameters for the SQL statement or stored procedure
RowSize	Specifies the amount of memory needed by the `OracleDataReader` internal cache to store one row of data
Site	Inherited from `Component`
Transaction	Specifies the `OracleTransaction` object in which the `OracleCommand` executes
UpdatedRowSource	Specifies how query command results are applied to the row being updated
XmlCommandType	Specifies the type of XML operation on the `OracleCommand`
XmlQueryProperties	Specifies the properties that are used when an XML document is created from the result set of a SQL query statement
XmlSaveProperties	Specifies the properties that are used when an XML document is used to save changes to the database

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

AddRowid

This property adds the `ROWID` as part of the select list.

Declaration

```
// C#
public bool AddRowid {get; set;}
```

Property Value

`bool`

Remarks

Default is `false`.

This `ROWID` column is hidden and is not accessible by the application. To gain access to the `ROWID`s of a table, the `ROWID` must explicitly be added to the select list without the use of this property.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- "LOB Support" on page 3-35 for further information on how this property used with LOBs

ArrayBindCount

This property specifies if the array binding feature is to be used and also specifies the number of array elements to be bound in the `OracleParameter Value` property.

Declaration

```
// C#
public int ArrayBindCount {get; set;}
```

Property Value

An `int` value that specifies number of array elements to be bound in the `OracleParameter` `Value` property.

Exceptions

`ArgumentException` - The `ArrayBindCount` value specified is invalid.

Remarks

Default = 0.

If `ArrayBindCount` is equal to `0`, array binding is not used; otherwise, array binding is used and `OracleParameter` `Value` property is interpreted as an array of values. The value of `ArrayBindCount` must be specified to use the array binding feature.

If neither `DbType` nor `OracleDbType` is set, it is strongly recommended that you set `ArrayBindCount` before setting the `OracleParameter` `Value` property so that inference of `DbType` and `OracleDbType` from `Value` can be correctly done.

Array binding is not used by default.

If the `XmlCommandType` property is set to any value other than `None`, this property is ignored.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- "Array Binding" on page 3-29
- "Value" on page 4-61

BindByName

This property specifies the binding method in the collection.

Declaration

```
// C#
public bool BindByName {get; set;}
```

Property Value

Returns `true` if the parameters are bound by name; returns `false` if the parameters are bound by position.

Remarks

Default = `false`.

`BindByName` is supported only for `OracleCommand.CommandType = CommandType.Text`, not for `OracleCommand.CommandType = CommandType.StoredProcedure`.

`BindByName` is ignored under the following conditions:

- The value of the `XmlCommandType` property is `Insert`, `Update`, or `Delete`.
- The value of the `XmlCommandType` property is `Query`, but there are no parameters set on the `OracleCommand`.

If the `XmlCommandType` property is `OracleXmlCommandType.Query` and any parameters are set on the `OracleCommand`, the `BindByName` property must be set to `true`. Otherwise, the following `OracleCommand` methods throw an `InvalidOperationException`.

- `ExecuteNonQuery`
- `ExecuteXmlReader`
- `ExecuteStream`
- `ExecuteToStream`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- "Array Binding" on page 3-29
- "Value" on page 4-61

CommandText

This property specifies the SQL statement or stored procedure to run against the Oracle database or the XML data used to store changes to the Oracle database.

Declaration

```
// C#
public string CommandText {get; set;}
```

Property Value

A `string`.

Implements

`IDbCommand`

Remarks

The default is an empty string.

When the `CommandType` property is set to `StoredProcedure`, the `CommandText` property is set to the name of the stored procedure. The command calls this stored procedure when an `Execute` method is called.

The effects of `XmlCommandType` values on `CommandText` are:

- `XmlCommandType` = `None`.

  `CommandType` property determines the contents of `CommandText`.

- `XmlCommandType` = `Query`.

  `CommandText` must be a SQL query. The SQL query should be a select statement. `CommandType` property is ignored.

- `XmlCommandType` property is `Insert`, `Update`, or `Delete`.

  `CommandText` must be an XML document. `CommandType` property is ignored.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

CommandType

This property specifies the command type that indicates how the `CommandText` property is to be interpreted.

Declaration

```
// C#
public System.Data.CommandType CommandType {final get; final set;}
```

Property Value

A `CommandType`.

Exceptions

`ArgumentException` - The value is not a valid `CommandType` such as: `CommandType.Text`, `CommandType.StoredProcedure`, `CommandType.TableDirect`.

Remarks

Default = `CommandType.Text`

If the value of the `XmlCommandType` property is not `None`, then the `CommandType` property is ignored.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

Connection

This property specifies the `OracleConnection` object that is used to identify the connection to execute a command.

Declaration

```
// C#
public OracleConnection Connection {get; set;}
```

Property Value

An `OracleConnection` object.

Implements

`IDbCommand`

Remarks

Default = `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

FetchSize

This property specifies the size of `OracleDataReader`'s internal cache to store result set data.

Declaration

```
// C#
public long FetchSize {get; set;}
```

Property Value

A `long` that specifies the size (in bytes) of the `OracleDataReader`'s internal cache.

Exceptions

`ArgumentException` - The `FetchSize` value specified is invalid.

Remarks

Default = 65536.

The `FetchSize` property is inherited by the `OracleDataReader` that is created by a command execution returning a result set. The `FetchSize` property on the `OracleDataReader` object determines the amount of data the `OracleDataReader` fetches into its internal cache for each server round-trip.

If the `XmlCommandType` property is set to any value other than `None`, this property is ignored.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- `OracleDataReader` "FetchSize" on page 4-122

InitialLOBFetchSize

This property specifies the amount that the OracleDataReader initially fetches for LOB columns.

Declaration

```
// C#
public int InitialLOBFetchSize  {get; set;}
```

Property Value

An `int` specifying the amount.

Exceptions

`ArgumentException` - The `InitialLOBFetchSize` value specified is invalid.

Remarks

The maximum value supported for `InitialLOBFetchSize` is `32767`. If this property is set to a higher value, the provider resets it to `32767`.

Default = 0.

The value of `InitialLOBFetchSize` specifies the initial amount of LOB data that is immediately fetched by the `OracleDataReader`. The property value specifies the number of characters for `CLOB` and `NCLOB` data and the number of bytes for `BLOB` data. To fetch more than the specified `InitialLOBFetchSize` amount, one of the following must be in the select list:

- primary key
- `ROWID`
- unique columns - (defined as a set of columns on which a unique constraint has been defined or a unique index has been created, where at least one of the columns in the set has a `NOT NULL` constraint defined on it)

The InitialLOBFetchSize value is used to determine the length of the LOB column data to fetch if LOB column is in the select list. If the select list does not contain a LOB column, the InitialLOBFetchSize value is ignored.

A primary key, a ROWID, or unique columns are not required if this property is set to 0.

If the InitialLOBFetchSize is set to a nonzero value, GetOracleBlob() and GetOracleClob() methods are disabled. BLOB and CLOB data are fetched by using GetBytes() and GetChars(), respectively.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- "Obtaining LOB Data" on page 3-15

InitialLONGFetchSize

This property specifies the amount that the OracleDataReader initially fetches for LONG and LONG RAW columns.

Declaration

```
// C#
public int InitialLONGFetchSize  {get; set;}
```

Property Value

An int specifying the amount.

Exceptions

ArgumentException - The InitialLONGFetchSize value specified is invalid.

Remarks

The maximum value supported for InitialLONGFetchSize is 32767. If this property is set to a higher value, the provider resets it to 32767.

The value of InitialLONGFetchSize specifies the initial amount of LONG or LONG RAW data that is immediately fetched by the OracleDataReader. The property value specifies the number of characters for LONG data and the number

of bytes for LONG RAW. To fetch more than the specified InitialLONGFetchSize amount, one of the following must be in the select list:

- primary key
- ROWID
- unique columns - (defined as a set of columns on which a unique constraint has been defined or a unique index has been created, where at least one of the columns in the set has a NOT NULL constraint defined on it)

The InitialLONGFetchSize value is used to determine the length of the LONG and LONG RAW column data to fetch if one of the two is in the select list. If the select list does not contain a LONG or a LONG RAW column, the InitialLONGFetchSize value is ignored.

Default = 0.

Setting this property to 0 defers the LONG and LONG RAW data retrieval entirely until the application specifically requests it.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- "Obtaining LONG and LONG RAW Data" on page 3-14 for further information

Parameters

This property specifies the parameters for the SQL statement or stored procedure.

Declaration

```
// C#
public OracleParameterCollection Parameters {get;}
```

Property Value

OracleParameterCollection

Implements

IDbCommand

Remarks

Default value = an empty collection

The number of the parameters in the collection must be equal to the number of parameter placeholders within the command text, or an error is raised.

If the command text does not contain any parameter tokens (such as, `:1`, `:2`), the values in the `Parameters` property are ignored.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

RowSize

This property specifies the amount of memory needed by the `OracleDataReader` internal cache to store one row of data.

Declaration

```
// C#
public long RowSize {get;}
```

Property Value

A `long` that indicates the amount of memory (in bytes) that an `OracleDataReader` needs to store one row of data for the executed query.

Remarks

Default value = `0`

The `RowSize` property is set to a nonzero value after the execution of a command that returns a result set. This property can be used at design time or dynamically during run-time, to set the `FetchSize`, based on number of rows. For example, to enable the `OracleDataReader` to fetch `N` rows for each server round-trip, the `OracleDataReader`'s `FetchSize` property can be set dynamically to `RowSize * N`. Note that for the `FetchSize` to take effect appropriately, it must be set after `OracleCommand.ExecuteReader()` but before `OracleDataReader.Read()`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- `OracleDataReader` "FetchSize" on page 4-19

Transaction

This property specifies the `OracleTransaction` object in which the `OracleCommand` executes.

Declaration

```
// C#
public OracleTransaction Transaction {get;}
```

Property Value

`OracleTransaction`

Implements

`IDbCommand`

Remarks

Default value = `null`

`Transaction` returns a reference to the transaction object associated with the `OracleCommand` connection object. Thus the command is executed in whatever transaction context its connection is currently in.

Note: When this property is accessed through an `IDbCommand` reference, its set accessor method is not operational.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

UpdatedRowSource

This property specifies how query command results are applied to the row to be updated.

Declaration

```
// C#
public System.Data.UpdateRowSource UpdatedRowSource {final get; final set;}
```

Property Value

An `UpdateRowSource`.

Implements

`IDbCommand`

Exceptions

`ArgumentException` - The `UpdateRowSource` value specified is invalid.

Remarks

Default = `UpdateRowSource.None` if the command is automatically generated.
Default = `UpdateRowSource.Both` if the command is not automatically generated.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

XmlCommandType

This property specifies the type of XML operation on the `OracleCommand`.

Declaration

```
// C#
public OracleXmlCommandType XmlCommandType {get; set;}
```

Property Value

An `OracleXmlCommandType`.

Remarks

Default value is `None`.

`XmlCommandType` values and usage:

- `None` - The `CommandType` property specifies the type of operation.
- `Query` - `CommandText` property must be set to a SQL select statement. The query is executed, and the results are returned as an XML document. The SQL select statement in the `CommandText` and the properties specified by the `XmlQueryProperties` property are used to perform the operation. The `CommandType` property is ignored.
- `Insert`, `Update`, or `Delete` - `CommandText` property is an XML document containing the changes to be made. The XML document in the `CommandText` and the properties specified by the `XmlSaveProperties` property are used to perform the operation. The `CommandType` property is ignored.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

XmlQueryProperties

This property specifies the properties that are used when an XML document is created from the result set of a SQL query statement.

Declaration

```
// C#
public OracleXmlQueryProperties XmlQueryProperties {get; set;}
```

Property Value

`OracleXmlQueryProperties`.

Remarks

When a new instance of `OracleCommand` is created, an instance of `OracleXmlQueryProperties` is automatically available on the `OracleCommand` instance through the `OracleCommand.XmlQueryProperties` property.

A new instance of `OracleXmlQueryProperties` can be assigned to an `OracleCommand` instance. Assigning an instance of

OracleXmlQueryProperties to the XmlQueryProperties of an OracleCommand instance creates a new instance of the given OracleXmlQueryProperties instance for the OracleCommand. This way each OracleCommand instance has its own OracleXmlQueryProperties instance.

Use the default constructor to get a new instance of OracleXmlQueryProperties.

Use the OracleXmlQueryProperties.Clone() method to get a copy of an OracleXmlQueryProperties instance.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

XmlSaveProperties

This property specifies the properties that are used when an XML document is used to save changes to the database.

Declaration

```
// C#
public OracleXmlSaveProperties XmlSaveProperties {get; set;}
```

Property Value

OracleXmlSaveProperties.

Remarks

When a new instance of OracleCommand is created, an instance of OracleXmlSaveProperties is automatically available on the OracleCommand instance through the OracleCommand.XmlSaveProperties property.

A new instance of OracleXmlSaveProperties can be assigned to an OracleCommand instance. Assigning an instance of OracleXmlSaveProperties to the XmlSaveProperties of an OracleCommand instance creates a new instance of the given OracleXmlSaveProperties instance for the OracleCommand. This way each OracleCommand instance has its own OracleXmlSaveProperties instance.

Use the default constructor to get a new instance of OracleXmlSaveProperties.

Use the `OracleXmlSaveProperties.Clone()` method to get a copy of an `OracleXmlSaveProperties` instance.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

OracleCommand Public Methods

`OracleCommand` public methods are listed in Table 4–7.

Table 4–7 OracleCommand Public Methods

Public Method	Description
`Cancel`	*Not Supported*
Clone	Creates a copy of `OracleCommand` object
`CreateObjRef`	Inherited from `MarshalByRefObject`
CreateParameter	Creates a new instance of `OracleParameter` class
`Dispose`	Inherited from `Component`
`Equals`	Inherited from `Object` (Overloaded)
ExecuteNonQuery	Executes a SQL statement or a command using the `XmlCommandType` and `CommandText` properties and returns the number of rows affected
ExecuteReader	Executes a command (Overloaded)
ExecuteScalar	Returns the first column of the first row in the result set returned by the query
ExecuteStream	Executes a command using the `XmlCommandType` and `CommandText` properties and returns the results in a new `Stream` object
ExecuteToStream	Executes a command using the `XmlCommandType` and `CommandText` properties and appends the results as an XML document to the existing `Stream`

Table 4–7 OracleCommand Public Methods (Cont.)

Public Method	Description
ExecuteXmlReader	Executes a command using the `XmlCommandType` and `CommandText` properties and returns the result as an XML document in a .NET `XmlTextReader` object
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
`Prepare`	*This method is a no-op*
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

Clone

This method creates a copy of an `OracleCommand` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleCommand` object.

Implements

`ICloneable`

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
//Need a proper casting for the return value when cloned
OracleCommand cmd_cloned = (OracleCommand) cmd.Clone();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

CreateParameter

This method creates a new instance of `OracleParameter` class.

Declaration

```
// C#
public OracleParameter CreateParameter();
```

Return Value

A new `OracleParameter` with default values.

Implements

`IDbCommand`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

ExecuteNonQuery

This method executes a SQL statement or a command using the `XmlCommandType` and `CommandText` properties and returns the number of rows affected.

Declaration

```
// C#
public int ExecuteNonQuery();
```

Return Value

The number of rows affected.

Implements

`IDbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

`ExecuteNonQuery` returns the number of rows affected, for the following:

- If the command is `UPDATE`, `INSERT`, or `DELETE` and the `XmlCommandType` property is set to `OracleXmlCommandType.None`.
- If the `XmlCommandType` property is set to `OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, `OracleXmlCommandType.Delete`.

For all other types of statements, the return value is `-1`.

`ExecuteNonQuery` is used for either of the following:

- catalog operations (for example, querying the structure of a database or creating database objects such as tables).
- changing the data in a database without using a `DataSet`, by executing `UPDATE`, `INSERT`, or `DELETE` statements.
- changing the data in a database using an XML document.

Although `ExecuteNonQuery` does not return any rows, it populates any output parameters or return values mapped to parameters with data.

If the `XmlCommandType` property is set to `OracleXmlCommandType.Query` then `ExecuteNonQuery` executes the select statement in the `CommandText` property, and if successful, returns `-1`. The XML document that is generated is discarded. This is useful for determining if the operation completes successfully without getting the XML document back as a result.

If the `XmlCommandType` property is set to `OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, or `OracleXmlCommandType.Delete`, then the value of the `CommandText` property is an XML document. `ExecuteNonQuery` saves the changes in that XML document to the table or view that is specified in the `XmlSaveProperties` property. The return value is the number of rows that are processed in the XML document. Also, each row in the XML document could affect

multiple rows in the database, but the return value is still the number of rows in the XML document.

Example

```
// C#
...
OracleConnection con = new OracleConnection("User Id=scott;Password=tiger;"+
         "Data Source=oracle");
OracleCommand cmd = new OracleCommand("update emp set sal = 3000" +
         "where empno=7934", con);
cmd.Connection.Open();
cmd.ExecuteNonQuery();
cmd.Dispose();
...
```

Requirements

For XML support, this method requires Oracle9*i* XML Developer's Kits (Oracle XDK) or higher, to be installed in the database. Oracle XDK can be downloaded from Oracle Technology Network (OTN).

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- http://otn.oracle.com/

ExecuteReader

`ExecuteReader` executes a command specified in the `CommandText`.

Overload List:

- ExecuteReader()

  This method executes a command specified in the `CommandText` and returns an `OracleDataReader` object.

- ExecuteReader(CommandBehavior)

  This method executes a command specified in the `CommandText` and returns an `OracleDataReader` object, using the specified `CommandBehavior` value.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

ExecuteReader()

This method executes a command specified in the `CommandText` and returns an `OracleDataReader` object.

Declaration

```
// C#
public OracleDataReader ExecuteReader();
```

Return Value

An `OracleDataReader`.

Implements

`IDbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

When the `CommandType` property is set to `CommandType.StoredProcedure`, the `CommandText` property should be set to the name of the stored procedure.

The command executes this stored procedure when you call `ExecuteReader()`. If parameters for the stored procedure consists of `REF CURSORs`, behavior differs depending on whether `ExecuteReader()` or `ExecuteNonQuery()` is called.

The value of `100` is used for the `FetchSize`. If `0` is specified, no rows are fetched. For further information, see "Obtaining LONG and LONG RAW Data" on page 3-14.

If the value of the `XmlCommandType` property is set to `OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, `OracleXmlCommandType.Delete`, or `OracleXmlCommandType.Query` then the `ExecuteReader` method throws an `InvalidOperationException`.

Example

```
// C#
...
OracleConnection con = new OracleConnection("User Id=scott;Password=tiger;"
       + "Data Source=oracle");
OracleCommand cmd = new OracleCommand("select ename from emp", con);
cmd.Connection.Open();
OracleDataReader reader = cmd.ExecuteReader();

while (reader.Read())
{
    Console.WriteLine("Employee Name : " + reader.GetString(0));
}

reader.Dispose();
cmd.Dispose();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- "OracleRefCursor Class" on page 5-624

ExecuteReader(CommandBehavior)

This method executes a command specified in the `CommandText` and returns an `OracleDataReader` object, using the specified behavior.

Declaration

```
// C#
public OracleDataReader ExecuteReader(CommandBehavior behavior);
```

Parameters

- *behavior*

  Specifies expected behavior.

Return Value

An `OracleDataReader`.

Implements

IDbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

Remarks

A description of the results and the effect on the database of the query command is indicated by the supplied *behavior* that specifies command behavior.

For valid CommandBehavior values and for the expected behavior of each CommandBehavior enumerated type, read the .NET Framework documentation.

When the CommandType property is set to CommandType.StoredProcedure, the CommandText property should be set to the name of the stored procedure. The command executes this stored procedure when ExecuteReader() is called.

If the stored procedure returns stored REF CURSORs, read the section on OracleRefCursors for more details. See "OracleRefCursor Class" on page 5-624.

The value of 100 is used for the FetchSize. If 0 is specified, no rows are fetched. For more information, see "Obtaining LONG and LONG RAW Data" on page 3-14.

If the value of the XmlCommandType property is set to OracleXmlCommandType.Insert, OracleXmlCommandType.Update, OracleXmlCommandType.Delete, or OracleXmlCommandType.Query then the ExecuteReader method throws an InvalidOperationException.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- "OracleRefCursor Class" on page 5-624

ExecuteScalar

This method executes the query using the connection, and returns the first column of the first row in the result set returned by the query.

Declaration

```
// C#
```

```
public object ExecuteScalar();
```

Return Value

An object which represents the value of the first row, first column.

Implements

`IDbCommand`

Exceptions

`InvalidOperationException` - The command cannot be executed.

Remarks

Extra columns or rows are ignored. `ExecuteScalar` retrieves a single value (for example, an aggregate value) from a database. This requires less code than using the `ExecuteReader()` method, and then performing the operations necessary to generate the single value using the data returned by an `OracleDataReader`.

If the query does not return any row, it returns `null`.

The `ExecuteScalar` method throws an `InvalidOperationException`, if the value of the `XmlCommandType` property is set to one of the following `OracleXmlCommandType` values: `Insert`, `Update`, `Delete`, `Query`.

Example

```
// C#
...
CmdObj.CommandText = "select count(*) from emp";
decimal count = (decimal) CmdObj.ExecuteScalar();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members

ExecuteStream

This method executes a command using the `XmlCommandType` and `CommandText` properties and returns the result as an XML document in a new `Stream` object.

Declaration

```
// C#
public Stream ExecuteStream();
```

Return Value

A `Stream`.

Remarks

The behavior of `ExecuteStream` varies depending on the `XmlCommandType` property value:

- `XmlCommandType` = `OracleXmlCommandType.None`

  `ExecuteStream` throws an `InvalidOperationException`.

- `XmlCommandType` = `OracleXmlCommandType.Query`

  `ExecuteStream` executes the select statement in the `CommandText` property, and if successful, returns an `OracleClob` object containing the XML document that was generated. `OracleClob` contains Unicode characters.

  If the SQL query does not return any rows, then `ExcecuteStream` returns an `OracleClob` object containing an empty XML document.

- `XmlCommandType` = `OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, or `OracleXmlCommandType.Delete`.

  The value of the `CommandText` property is an XML document. `ExecuteStream` saves the data in that XML document to the table or view that is specified in the `XmlSaveProperties` property and an empty `OracleClob` is returned.

Requirements

For database releases 8.1.7 and 9.0.1 only: This method requires Oracle XML Developer's Kit (Oracle XDK) release 9.2 or higher to be installed on the database. Oracle XDK can be downloaded from Oracle Technology Network (OTN).

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- *Oracle XML DB Developer's Guide*
- http://otn.oracle.com/

ExecuteToStream

This method executes a command using the `XmlCommandType` and `CommandText` properties and appends the result as an XML document to the existing `Stream` provided by the application.

Declaration

```
// C#
public void ExecuteToStream(Stream outputStream);
```

Parameters

- *`outputStream`*

  A `Stream`.

Remarks

The behavior of `ExecuteToStream` varies depending on the `XmlCommandType` property value:

- `XmlCommandType` = `OracleXmlCommandType.None`

  `ExecuteToStream` throws an `InvalidOperationException`.

- `XmlCommandType` = `OracleXmlCommandType.Query`

  `ExecuteToStream` executes the select statement in the `CommandText` property, and if successful, appends the XML document that was generated to the given `Stream`.

  If the SQL query does not return any rows, then nothing is appended to the given `Stream`. The character set of the appended data is Unicode.

- `XmlCommandType` = `OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, or `OracleXmlCommandType.Delete`

The value of the `CommandText` property is an XML document. `ExecuteToStream` saves the changes in that XML document to the table or view that is specified in the `XmlSaveProperties` property. Nothing is appended to the given `Stream`.

Requirements

For database releases 8.1.7 and 9.0.1 only: This method requires Oracle XML Developer's Kit (Oracle XDK) release 9.2 or higher to be installed on the database. Oracle XDK can be downloaded from Oracle Technology Network (OTN).

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- *Oracle XML DB Developer's Guide*
- http://otn.oracle.com/

ExecuteXmlReader

This method executes the command using the `XmlCommandType` and `CommandText` properties and returns the result as an XML document in a .NET `XmlTextReader` object.

Declaration

```
// C#
public XmlReader ExecuteXmlReader();
```

Return Value

An `XmlReader`.

Remarks

The behavior of `ExecuteXmlReader` varies depending on the `XmlCommandType` property value:

- `XmlCommandType` = `OracleXmlCommandType.None`

  `ExecuteStream` throws an `InvalidOperationException`.

- `XmlCommandType` = `OracleXmlCommandType.Query`

`ExecuteXmlReader` executes the select statement in the `CommandText` property, and if successful, returns a .NET `XmlTextReader` object containing the XML document that was generated.

If the XML document is empty, which can happen if the SQL query does not return any rows, then an empty .NET `XmlTextReader` object is returned.

- `XmlCommandType` = `OracleXmlCommandType.Insert`, `OracleXmlCommandType.Update`, or `OracleXmlCommandType.Delete`.

  The value of the `CommandText` property is an XML document, and `ExecuteXmlReader` saves the changes in that XML document to the table or view that is specified in the `XmlSaveProperties` property. An empty .NET `XmlTextReader` object is returned.

Requirements

For database releases 8.1.7 and 9.0.1 only: This method requires Oracle XML Developer's Kit (Oracle XDK) release 9.2 or higher to be installed on the database. Oracle XDK can be downloaded from Oracle Technology Network (OTN).

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommand Class
- OracleCommand Members
- *Oracle XML DB Developer's Guide*
- http://otn.oracle.com/

OracleCommandBuilder Class

An `OracleCommandBuilder` object provides automatic SQL generation for the `OracleDataAdapter` when updates are made to the database.

Class Inheritance

```
Object
  MarshalByRefObject
    Component
      OracleCommandBuilder
```

Declaration

```
// C#
public sealed class OracleCommandBuilder : Component
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

`OracleCommandBuilder` automatically generates SQL statements for single-table updates when the `SelectCommand` property of the `OracleDataAdapter` is set. An exception is thrown if the `DataSet` contains multiple tables. The `OracleCommandBuilder` registers itself as a listener for `RowUpdating` events whenever its `DataAdapter` property is set. Only one `OracleDataAdapter` object and one `OracleCommandBuilder` object can be associated with each other at one time.

To generate `INSERT`, `UPDATE`, or `DELETE` statements, the `OracleCommandBuilder` uses `ExtendedProperties` within the `DataSet` to retrieve a required set of metadata. If the `SelectCommand` is changed after the metadata is retrieved (for example, after the first update), the `RefreshSchema` method should be called to update the metadata.

`OracleCommandBuilder` first looks for the metadata from the `ExtendedProperties` of the `DataSet`; if the metadata is not available, `OracleCommandBuilder` uses the `SelectCommand` property of the `OracleDataAdapter` to retrieve the metadata.

Example

The `OracleCommandBuilder` examples in this section are based on the `EMPINFO` table which is defined as follows:

```
CREATE TABLE empInfo (
 empno NUMBER(4) PRIMARY KEY,
 empName VARCHAR2(20) NOT NULL,
 hiredate DATE,
 salary NUMBER(7,2),
 jobDescription Clob,
 byteCodes BLOB
);
```

The `EMPINFO` table has the following values:

```
EMPNO    EMPNAME    HIREDATE    SALARY     JOBDESCRIPTION  BYTECODES
                                                          (Hex Values)
=====    =======    ========    ======     ==============  ============

    1    KING       01-MAY-81   12345.67   SOFTWARE ENGR   {0x12, 0x34}
    2    SCOTT      01-SEP-75   34567.89   MANAGER         {0x56, 0x78}
    3    BLAKE      01-OCT-90   9999.12    TRANSPORT       {0x23, 0x45}
    4    SMITH      NULL        NULL       NULL             NULL
```

The following example uses the `OracleCommandBuilder` object to create the `UpdateCommand` for the `OracleDataAdapter` object when `OracleDataAdapter.Update()` is called.

```
// C#
public static void BuilderUpdate(string connStr)
{
  string cmdStr = "SELECT EMPNO, EMPNAME, JOBDESCRIPTION FROM EMPINFO";

  //create the adapter with the selectCommand txt and the
  //connection string
  OracleDataAdapter adapter = new OracleDataAdapter(cmdStr, connStr);

  //get the connection from the adapter
  OracleConnection connection = adapter.SelectCommand.Connection;

  //create the builder for the adapter to automatically generate
  //the Command when needed
  OracleCommandBuilder builder = new OracleCommandBuilder(adapter);
```

```
  //Create and fill the DataSet using the EMPINFO
  DataSet dataset = new DataSet();
  adapter.Fill(dataset, "EMPINFO");

  //Get the EMPINFO table from the dataset
  DataTable table = dataset.Tables["EMPINFO"];

  //Get the first row from the EMPINFO table
  DataRow row0 = table.Rows[0];

  //update the job description in the first row
  row0["JOBDESCRIPTION"] = "MANAGER";

  //Now update the EMPINFO using the adapter, the job description
  //of 'KING' is changed to 'MANAGER'
  //The OracleCommandBuilder will create the UpdateCommand for the
  //adapter to update the EMPINFO table
  adapter.Update(dataset, "EMPINFO");

}
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Members
- OracleCommandBuilder Constructors
- OracleCommandBuilder Static Methods
- OracleCommandBuilder Properties
- OracleCommandBuilder Public Methods
- OracleCommandBuilder Events
- OracleCommandBuilder Event Delegates

OracleCommandBuilder Members

`OracleCommandBuilder` members are listed in the following tables:

OracleCommandBuilder Constructors

`OracleCommandBuilder` constructors are listed in Table 4–8.

Table 4–8 OracleCommandBuilder Constructors

Constructor	Description
OracleCommandBuilder Constructors	Instantiates a new instance of `OracleCommandBuilder` class (Overloaded)

OracleCommandBuilder Static Methods

`OracleCommandBuilder` static methods are listed in Table 4–9.

Table 4–9 OracleCommandBuilder Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleCommandBuilder Properties

`OracleCommandBuilder` properties are listed in Table 4–10.

Table 4–10 OracleCommandBuilder Properties

Name	Description
`Container`	Inherited from `Component`
DataAdapter	Indicates the `OracleDataAdapter` for which the SQL statements are generated
CaseSensitive	Indicates whether or not double quotes are used around Oracle object names when generating SQL statements
`Site`	Inherited from `Component`

OracleCommandBuilder Public Methods

`OracleCommandBuilder` public methods are listed in Table 4–11.

Table 4–11 OracleCommandBuilder Public Methods

Public Method	Description
CreateObjRef	Inherited from MarshalByRefObject
Dispose	Inherited from Component
Equals	Inherited from Object (Overloaded)
GetDeleteCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform deletions on the database
GetHashCode	Inherited from Object
GetInsertCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform insertions on the database
GetLifetimeService	Inherited from MarshalByRefObject
GetType	Inherited from Object
GetUpdateCommand	Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform updates on the database
InitializeLifetimeService	Inherited from MarshalByRefObject
RefreshSchema	Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements
ToString	Inherited from Object

OracleCommandBuilder Events

OracleCommandBuilder events are listed in Table 4–12.

Table 4–12 OracleCommandBuilder Events

Event Name	Description
Disposed	Inherited from Component

OracleCommandBuilder Event Delegates

OracleCommandBuilder event delegates are listed in Table 4–13.

Table 4–13 OracleCommandBuilder Event Delegates

Event Delegate Name	Description
`EventHandler`	Inherited from `Component`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class

OracleCommandBuilder Constructors

`OracleCommandBuilder` constructors create new instances of the `OracleCommandBuilder` class.

Overload List:

- OracleCommandBuilder()

  This constructor creates an instance of the `OracleCommandBuilder` class.

- OracleCommandBuilder(OracleDataAdapter)

  This constructor creates an instance of the `OracleCommandBuilder` class and sets the `DataAdapter` property to the provided `OracleDataAdapter` object.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

OracleCommandBuilder()

This constructor creates an instance of the `OracleCommandBuilder` class.

Declaration

```
// C#
public OracleCommandBuilder();
```

Remarks

Default constructor.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

OracleCommandBuilder(OracleDataAdapter)

This constructor creates an instance of the `OracleCommandBuilder` class and sets the `DataAdapter` property to the provided `OracleDataAdapter` object.

Declaration

```
// C#
public OracleCommandBuilder(OracleDataAdapter da);
```

Parameters

- *da*

  The `OracleDataAdapter` object provided.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

OracleCommandBuilder Static Methods

`OracleCommandBuilder` properties are listed in Table 4–14.

Table 4–14 OracleCommandBuilder Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

OracleCommandBuilder Properties

`OracleCommandBuilder` properties are listed in Table 4–15.

Table 4–15 OracleCommandBuilder Properties

Name	Description
`Container`	Inherited from `Component`
DataAdapter	Indicates the `OracleDataAdapter` for which the SQL statements are generated
CaseSensitive	Indicates whether or not double quotes are used around Oracle object names when generating SQL statements
`Site`	Inherited from `Component`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

DataAdapter

This property indicates the `OracleDataAdapter` for which the SQL statements are generated.

Declaration

```
// C#
OracleDataAdapter DataAdapter{get; set;}
```

Property Value

`OracleDataAdapter`

Remarks

Default = `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

CaseSensitive

This property indicates whether or not double quotes are used around Oracle object names (for example, tables or columns) when generating SQL statements.

Declaration

```
// C#
bool CaseSensitive {get; set;}
```

Property Value

A `bool` that indicates whether or not double quotes are used.

Remarks

Default = `false`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

OracleCommandBuilder Public Methods

`OracleCommandBuilder` public methods are listed in Table 4–16.

Table 4–16 OracleCommandBuilder Public Methods

Public Method	Description
`CreateObjRef`	Inherited from `MarshalByRefObject`
`Dispose`	Inherited from `Component`
`Equals`	Inherited from `Object` (Overloaded)

Table 4–16 OracleCommandBuilder Public Methods (Cont.)

Public Method	Description
GetDeleteCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database
`GetHashCode`	Inherited from `Object`
GetInsertCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
GetUpdateCommand	Gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
RefreshSchema	Refreshes the database schema information used to generate `INSERT`, `UPDATE`, or `DELETE` statements
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

GetDeleteCommand

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform deletions on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#
public OracleCommand GetDeleteCommand();
```

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

GetInsertCommand

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform insertions on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#
public OracleCommand GetInsertCommand();
```

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

GetUpdateCommand

This method gets the automatically generated `OracleCommand` object that has the SQL statement (`CommandText`) perform updates on the database when an application calls `Update()` on the `OracleDataAdapter`.

Declaration

```
// C#
public OracleCommand GetUpdateCommand();
```

Return Value

An `OracleCommand`.

Exceptions

`ObjectDisposedException` - The `OracleCommandBuilder` object is already disposed.

`InvalidOperationException` - Either the `SelectCommand` or the `DataAdapter` property is null, or the primary key cannot be retrieved from the `SelectCommand` property of the `OracleDataAdapter`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

RefreshSchema

This method refreshes the database schema information used to generate `INSERT`, `UPDATE`, or `DELETE` statements.

Declaration

```
// C#
public void RefreshSchema();
```

Remarks

An application should call `RefreshSchema` whenever the `SelectCommand` value of the `OracleDataAdapter` changes.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

OracleCommandBuilder Events

`OracleCommandBuilder` events are listed in Table 4–17.

Table 4–17 OracleCommandBuilder Events

Event Name	Description
`Disposed`	Inherited from `Component`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

OracleCommandBuilder Event Delegates

`OracleCommandBuilder` event delegates are listed in Table 4–18.

Table 4–18 OracleCommandBuilder Event Delegates

Event Delegate Name	Description
`EventHandler`	Inherited from `Component`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleCommandBuilder Class
- OracleCommandBuilder Members

OracleConnection Class

An `OracleConnection` object represents a connection to an Oracle database.

Class Inheritance

```
Object
  MarshalByRefObject
    Component
      OracleConnection
```

Declaration

```
// C#
public sealed class OracleConnection : Component,
   IDbConnection, ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
//  Uses connection to create and return an OracleCommand object.
...
string ConStr = "User Id=myschema;Password=mypassword;" +
      "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();
OracleCommand cmd = con.CreateCommand();

cmd.CommandText = "insert into mytable values (99, 'foo')";
cmd.CommandType = CommandType.Text;
cmd.ExecuteNonQuery();
...
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Members
- OracleConnection Constructors
- OracleConnection Static Methods
- OracleConnection Properties
- OracleConnection Public Methods
- OracleConnection Events
- OracleConnection Event Delegates

OracleConnection Members

`OracleConnection` members are listed in the following tables:

OracleConnection Constructors

`OracleConnection` constructors are listed in Table 4–19.

Table 4–19 OracleConnection Constructors

Constructor	Description
OracleConnection Constructors	Instantiates a new instance of the `OracleConnection` class (Overloaded)

OracleConnection Static Methods

`OracleConnection` static methods are listed in Table 4–20.

Table 4–20 OracleConnection Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleConnection Properties

`OracleConnection` properties are listed in Table 4–21

Table 4–21 OracleConnection Properties

Name	Description
ConnectionString	Specifies connection information used to connect to an Oracle database
ConnectionTimeout	Specifies the maximum amount of time that the `Open()` method can take to obtain a pooled connection before terminating the request
`Container`	Inherited from `Component`
DataSource	Specifies the Oracle Net Service Name (also known as TNS alias) that identifies an Oracle database instance
ServerVersion	Specifies the version number of the Oracle database to which the `OracleConnection` has established a connection
`Site`	Inherited from `Component`
State	Specifies the current state of the connection

OracleConnection Public Methods

`OracleConnection` public methods are listed in Table 4–22.

Table 4–22 OracleConnection Public Methods

Public Method	Description
BeginTransaction	Begins a local transaction (Overloaded)
`ChangeDatabase`	*Not Supported*
Clone	Creates a copy of an `OracleConnection` object
Close	Closes the database connection
CreateCommand	Creates and returns an `OracleCommand` object associated with the `OracleConnection` object
`CreateObjRef`	Inherited from `MarshalByRefObject`
`Dispose`	Inherited from `Component`
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`

Table 4–22 OracleConnection Public Methods (Cont.)

Public Method	Description
GetSessionInfo	Returns or refreshes the property values of the `OracleGlobalization` object that represents the globalization settings of the session (Overloaded)
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
Open	Opens a database connection with the property settings specified by the `ConnectionString`
OpenWithNewPassword	Opens a new connection with the new password
SetSessionInfo	Alters the session's globalization settings with the property values provided by the `OracleGlobalization` object
`ToString`	Inherited from `Object`

OracleConnection Events

`OracleConnection` events are listed in Table 4–23.

Table 4–23 OracleConnection Events

Event Name	Description
`Disposed`	Inherited from `Component`
Failover	An event that is triggered when an Oracle failover occurs
InfoMessage	An event that is triggered for any message or warning sent by the database
StateChange	An event that is triggered when the connection state changes

OracleConnection Event Delegates

`OracleConnection` event delegates are listed in Table 4–24.

Table 4–24 OracleConnection Event Delegates

Event Delegate Name	Description
OracleFailoverEventHandler	An event delegate that handles the `Failover` event
OracleInfoMessageEventHandler	An event delegate that handles the `InfoMessage` event
StateChangeEventHandler	An event delegate that handles the `StateChange` event

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class

OracleConnection Constructors

`OracleConnection` constructors instantiate new instances of the `OracleConnection` class.

Overload List:

- OracleConnection()

  This constructor instantiates a new instance of the `OracleConnection` class using default property values.

- OracleConnection(String)

  This constructor instantiates a new instance of the `OracleConnection` class with the provided connection string.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

OracleConnection()

This constructor instantiates a new instance of the `OracleConnection` class using default property values.

Declaration

```
// C#
public OracleConnection();
```

Remarks

The properties for `OracleConnection` are set to the following default values:

- `ConnectionString` = empty string
- `ConnectionTimeout` = `15`
- `DataSource` = empty string
- `ServerVersion` = empty string

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

OracleConnection(String)

This constructor instantiates a new instance of the `OracleConnection` class with the provided connection string.

Declaration

```
// C#
public OracleConnection(String connectionString);
```

Parameters

- *`connectionString`*

  The connection information used to connect to the Oracle database.

Remarks

The `ConnectionString` property is set to the supplied *`connectionString`*. The `ConnectionString` property is parsed and an exception is thrown if it contains invalid connection string attributes or attribute values.

The properties of the `OracleConnection` object default to the following values unless they are set by the connection string:

- `ConnectionString` = empty string

- `ConnectionTimeout` = 15
- `DataSource` = empty string
- `ServerVersion` = empty string

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

OracleConnection Static Methods

`OracleConnection` static methods are listed in Table 4–25.

Table 4–25 OracleConnection Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

OracleConnection Properties

`OracleConnection` properties are listed in Table 4–26

Table 4–26 OracleConnection Properties

Name	Description
ConnectionString	Specifies connection information used to connect to an Oracle database
ConnectionTimeout	Specifies the maximum amount of time that the `Open()` method can take to obtain a pooled connection before terminating the request
`Container`	Inherited from `Component`

Table 4–26 OracleConnection Properties (Cont.)

Name	Description
DataSource	Specifies the Oracle Net Service Name (also known as TNS alias) that identifies an Oracle database instance
ServerVersion	Specifies the version number of the Oracle database to which the `OracleConnection` has established a connection
`Site`	Inherited from `Component`
State	Specifies the current state of the connection

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

ConnectionString

This property specifies connection information used to connect to an Oracle database.

Declaration

```
// C#
public string ConnectionString{get; set;}
```

Property Value

If the connection string is supplied through the constructor, this property is set to that string.

Implements

`IDbConnection`

Exceptions

`ArgumentException` - An invalid syntax is specified for the connection string.

`InvalidOperationException` - `ConnectionString` is being set while the connection is open.

Remarks

The default value is an empty string.

`ConnectionString` must be a string of attribute name and value pairings, separated by a semi-colon, for example:

```
// C#
OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=MYSCHEMA;Password=MYPASSWORD;" +
   "Data Source=Oracle";
```

If the `ConnectionString` is not in a proper format, an exception is thrown. All spaces are ignored unless they are within double quotes.

When the `ConnectionString` property is set, the `OracleConnection` object immediately parses the string for errors. An `ArgumentException` is thrown if the `ConnectionString` contains invalid attributes or invalid values. Attribute values for `User Id`, `Password`, `Proxy User Id`, `Proxy Password`, and `Data Source` (if provided) are not validated until the `Open` method is called.

The connection must be closed to set the `ConnectionString` property. When the `ConnectionString` property is reset, all previously set values are reinitialized to their default values before the new values are applied.

The Oracle database supports case-sensitive user names. To connect as a user whose name is of mixed case, for example, "`MySchema`", the `User Id` attribute value must be surrounded by double quotes, as follows:

```
// C#
OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=\"MySchema\";Password=MYPASSWORD;" +
    "Data Source=Oracle";
```

However, if the Oracle user name is all upper case, the `User Id` connection string attribute can be set to that user name without the use of the double quotes since `User Id`s that are not doubled-quoted are converted to all upper case when connecting. Single quotes are not supported.

See Also: "Example" on page 4-66 for a complete example

If a connection string attribute is set more than once, the last setting takes effect and no exceptions are thrown.

Boolean connection string attributes can be set to either `true`, `false`, `yes`, or `no`.

Supported connection string attributes:

Table 4–27 lists the supported connection string attributes.

Table 4–27 Supported Connection String Attributes

Connection String Attribute	Default value	Description
`Connection Lifetime`	`0`	Maximum life time (in seconds) of the connection This attribute specifies the lifetime of the connection in seconds. Before the `Connection` is placed back into the pool, the lifetime of the connection is checked. If the lifetime of the connection exceeds this property value, the connection is closed and disposed. If this property value is `0`, the connection lifetime is never checked. Connections that have exceeded their lifetimes are not closed and disposed of, if doing so brings the number of connection in the pool below the `Min Pool Size`.
`Connection Timeout`	`15`	Maximum time (in seconds) to wait for a free connection from the pool This attribute specifies the maximum amount of time (in seconds) that the `Open()` method can take to obtain a pooled connection before it terminates the request. This value comes into effect only if no free connection is available from the connection pool and the `Max Pool Size` is reached. If a free connection is not available within the specified time, an exception is thrown. `Connection Timeout` does not limit the time required to open new connections. This attribute value takes effect for pooled connection requests and not for new connection requests.
`Data Source`	empty string	Oracle Net Service Name that identifies the database to connect to This attribute specifies the Oracle Net Service Name (formerly known as TNS alias) that identifies an Oracle database instance. This attribute must be set to connect to a remote database.
`DBA Privilege`	empty string	Administrative privileges `SYSDBA` or `SYSOPER` This connection string attribute only accepts `SYSDBA` or `SYSOPER` as the attribute value. It is case insensitive.
`Decr Pool Size`	`1`	Number of connections that are closed when an excessive amount of established connections are unused.

Table 4–27 Supported Connection String Attributes (Cont.)

Connection String Attribute	Default value	Description
		This connection string attribute controls the maximum number of unused connections that are closed when the pool regulator makes periodic checks. The regulator thread is spawned every 3 minutes and closes up to `Decr Pool Size` amount of pooled connections if they are not used. The pool regulator never takes the total number of connections below the `Min Pool Size` by closing pooled connections.
`Enlist`	`true`	Serviced Components automatically enlist in distributed transactions If this attribute is set to `true`, the connection is automatically enlisted in the thread's transaction context. If this attribute is `false`, no enlistments are made. This attribute can be set to either `true`, `false`, `yes`, or `no`.
`Incr Pool Size`	`5`	Number of connections established when all connections in pool are used This connection string attribute determines the number of new connections that are established when a pooled connection is requested, but no unused connections are available and `Max Pool Size` is not reached. If new connections have been created for a pool, the regulator thread skips a cycle and does not have an opportunity to close any connections for 6 minutes. Note, however, that some connections can be still be closed during this time if their lifetime has been exceeded.
`Max Pool Size`	`100`	Maximum number of connections in a pool This attribute specifies the maximum number of connections allowed in the particular pool used by that `OracleConnection`. Simply changing this attribute in the connection string does not change the `Max Pool Size` restriction on a currently existing pool. Doing so simply creates a new pool with a different `Max Pool Size` restriction. This attribute must be set to a value greater than the `Min Pool Size`. This value is ignored unless `Pooling` is turned on.

Table 4–27 Supported Connection String Attributes (Cont.)

Connection String Attribute	Default value	Description
`Min Pool Size`	`1`	Minimum number of connections in a pool This attribute specifies the minimum number of connections to be maintained by the pool during its entire lifetime. Simply changing this attribute in the connection string does not change the `Min Pool Size` restriction on a currently existing pool. Doing so simply creates a new pool with a different `Min Pool Size` restriction. This value is ignored unless `Pooling` is turned on.
`Password`	empty string	Password for the user specified by `User Id` This attribute specifies an Oracle user's password. Password is case insensitive.
`Persist Security Info`	`false`	Enables or disables the retrieval of password in the connection string If this attribute is set to `false`, the `Password` value setting is not returned when the application requests the `ConnectionString` after the connection is successfully opened by the `Open()` method. This attribute can be set to either `true`, `false`, `yes`, or `no`.
`Pooling`	`true`	Enables or disables connection pooling This attribute specifies whether connection pooling is to be used. Pools are created using an attribute value matching algorithm. This means that connection strings which only differ in the number of spaces in the connection string use the same pool. If two connection strings are identical except that one sets an attribute to a default value while the other does not set that attribute, both requests obtain connections from the same pool. This attribute can be set to either `true`, `false`, `yes`, or `no`.
`Proxy User Id`	empty string	User name of the proxy user This connection string attribute specifies the middle-tier user, or the proxy user, who establishes a connection on behalf of a client user specified by the `User Id` attribute. ODP.NET attempts to establish a proxy connection if either the `Proxy User Id` or the `Proxy Password` attribute is set to a non-empty string.

Table 4–27 Supported Connection String Attributes (Cont.)

Connection String Attribute	Default value	Description
		For the proxy user to connect to an Oracle database using operating system authentication, the `Proxy User Id` must be set to "/". The `Proxy Password` is ignored in this case. The `User Id` cannot be set to "/" when establishing proxy connections. The case of this attribute value is preserved if it is surrounded by double quotes.
`Proxy Password`	empty string	Password of the proxy user This connection string attribute specifies the password of the middle-tier user or the proxy user. This user establishes a connection on behalf of a client user specified by the `User Id` attribute. ODP.NET attempts to establish a proxy connection if either the `Proxy User Id` or the `Proxy Password` attribute is set to a non-empty string.
`User Id`	empty string	Oracle user name This attribute specifies the Oracle user name. The case of this attribute value is preserved if it is surrounded by double quotes. For the user to connect to an Oracle database using operating system authentication, set the `User Id` to "/". Any `Password` attribute setting is ignored in this case.

Example

This code example shows that the case of the `User Id` attribute value is not preserved unless it is surrounded by double quotes. The example also demonstrates when connection pools are created and when connections are drawn from the connection pool.

```
// C#
// Assume users "MYSCHEMA"and "MySchema" exist in the database
...
OracleConnection con1 = new OracleConnection();
con1.ConnectionString = "User Id=myschema;Password=mypassword;" +
     "Data Source=oracle;";
con1.Open();  // Attempts to connect as "MYSCHEMA/MYPASSWORD"
            // A new connection is created; A new Connection Pool X is created
con1.Dispose(); // Connection is placed back into Pool X

OracleConnection con2 = new OracleConnection();
con2.ConnectionString = "User Id=MySchema;Password=MyPassword;" +
     "Data Source=oracle;";
```

```
con2.Open(); // Attempts to connect as "MYSCHEMA/MYPASSWORD" A connection is
             // obtained from Pool X; A new connection is NOT created.
con2.Dispose(); // Connection is placed back into Pool X

OracleConnection con3 = new OracleConnection();
con3.ConnectionString = "User Id=\"MYSCHEMA\";Password=MYPASSWORD;" +
     "Data Source=oracle;";
con3.Open(); // Attempts to connect as "MYSCHEMA/MYPASSWORD" A connection is
             // obtained from Pool X; A new connection is NOT created.
con3.Dispose(); // Connection is placed back into Pool X

OracleConnection con4 = new OracleConnection();
con4.ConnectionString = "User Id=\"MySchema\";Password=mypassword;" +
     "Data Source=oracle;";
con4.Open(); // Attempts to connect as "MySchema/MYPASSWORD"
             // A new connection is created; A new Connection Pool Y is created
con4.Dispose(); // Connection is placed back into Pool Y

OracleConnection con5 = new OracleConnection();
con5.ConnectionString = "User Id=MySchema;Password=mypassword;" +
     "Data Source=oracle;    ";
con5.Open(); // Attempts to connect as "MYSCHEMA/MYPASSWORD"
             // A connection is obtained from Connection Pool X
             // Extra spaces in the connection string do not force creation
             // of a new pool
con5.Dispose(); // Connection is placed back into Pool X

OracleConnection con6 = new OracleConnection();
con6.ConnectionString = "User Id=MySchema;Password=mypassword;" +
     "Data Source=oracle;Pooling=true;";
con6.Open(); // Attempts to connect as "MYSCHEMA/MYPASSWORD"
             // A connection is obtained from Connection Pool X. "Pooling=true"
             // in the connection string does not force creation of a new pool
             // since the initial connection was established using the default
             // value of "Pooling=true".  Note that even if the connection
             // string had "POOLING=Yes", a new connection pool will not be
             // created since they both enable pooling.  The same rule applies
             // to other connection string attributes as well.
con6.Dispose(); // Connection is placed back into Pool X
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

ConnectionTimeout

This property specifies the maximum amount of time that the `Open()` method can take to obtain a pooled connection before terminating the request.

Declaration

```
// C#
public int ConnectionTimeout {get;}
```

Property Value

The maximum time allowed for a pooled connection request, in seconds.

Implements

`IDbConnection`

Remarks

The default value is `15`.

Setting this property to `0` allows the pooled connection request to wait for a free connection without a time limit. The timeout takes effect only for pooled connection requests and not for new connection requests.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

DataSource

This property specifies the Oracle Net Service Name (formerly known as TNS alias) that identifies an Oracle database instance.

Declaration

```
// C#
```

```
public string DataSource {get;}
```

Property Value

The Oracle Net Service Name.

Remarks

The default value of this property is an empty string

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

ServerVersion

This property specifies the version number of the Oracle database to which the `OracleConnection` has established a connection.

Declaration

```
// C#
public string ServerVersion {get;}
```

Property Value

The version of the Oracle database, for example "9.2.0.1.0."

Exceptions

`InvalidOperationException` - The connection is closed.

Remarks

The default is an empty string.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

State

This property specifies the current state of the connection.

Declaration

```
// C#
public ConnectionState State {get;}
```

Property Value

The `ConnectionState` of the connection.

Implements

`IDbConnection`

Remarks

ODP.NET supports `ConnectionState.Closed` and `ConnectionState.Open` for this property. The default value is `ConnectionState.Closed`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

OracleConnection Public Methods

`OracleConnection` public methods are listed in Table 4–28.

Table 4–28 OracleConnection Public Methods

Public Method	Description
BeginTransaction	Begins a local transaction (Overloaded)
`ChangeDatabase`	*Not Supported*
Clone	Creates a copy of an `OracleConnection` object
Close	Closes the database connection
CreateCommand	Creates and returns an `OracleCommand` object associated with the `OracleConnection` object
`CreateObjRef`	Inherited from `MarshalByRefObject`
`Dispose`	Inherited from `Component`

Table 4–28 OracleConnection Public Methods (Cont.)

Public Method	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
GetSessionInfo	Returns or refreshes the property values of the `OracleGlobalization` object that represents the globalization settings of the session (Overloaded)
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
Open	Opens a database connection with the property settings specified by the `ConnectionString`
OpenWithNewPassword	Opens a new connection with the new password
SetSessionInfo	Alters the session's globalization settings with the property values provided by the `OracleGlobalization` object
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

BeginTransaction

`BeginTransaction` methods begin local transactions.

Overload List

- BeginTransaction()

  This method begins a local transaction.

- BeginTransaction(IsolationLevel)

  This method begins a local transaction with the specified isolation level.

BeginTransaction()

This method begins a local transaction.

Declaration

```
// C#
public OracleTransaction BeginTransaction();
```

Return Value

An `OracleTransaction` object representing the new transaction.

Implements

`IDbConnection`

Exceptions

`InvalidOperationException` - A transaction has already been started.

Remarks

The transaction is created with its isolation level set to its default value of `IsolationLevel.ReadCommitted`. All further operations related to the transaction must be performed on the returned `OracleTransaction` object.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

BeginTransaction(IsolationLevel)

This method begins a local transaction with the specified isolation level.

Declaration

```
// C#
public OracleTransaction BeginTransaction(IsolationLevel isolationLevel);
```

Parameters

- *`isolationLevel`*

  The isolation level for the new transaction.

Return Value

An `OracleTransaction` object representing the new transaction.

Implements

`IDbConnection`

Exceptions

`InvalidOperationException` - A transaction has already been started.

`ArgumentException` - The `isolationLevel` specified is invalid.

Remarks

The following two isolation levels are supported:

- `IsolationLevel.ReadCommitted`
- `IsolationLevel.Serializable`

Requesting other isolation levels causes an exception.

Example

```
// C#
// Starts a transaction and inserts one record. If insert fails, rolls back
// the transaction. Otherwise, commits the transaction.

...
string ConStr = "User Id=myschema;Password=mypassword;" +
        "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Create an OracleCommand object using the connection object
OracleCommand cmd = new OracleCommand("", con);

// Start a transaction
OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

try
{
   cmd.CommandText = "insert into mytable values (99, 'foo')";
   cmd.CommandType = CommandType.Text;
   cmd.ExecuteNonQuery();
   txn.Commit();
```

```
   Console.WriteLine("Both records are inserted into the database table.");
}
catch(Exception e)
{
   txn.Rollback();
   Console.WriteLine("Neither record was inserted into the database table.");
 }
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

Clone

This method creates a copy of an `OracleConnection` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleConnection` object.

Implements

`ICloneable`

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
OracleConnection con = new OracleConnection(ConStr);
con.Open();
...

//Need a proper casting for the return value when cloned
```

```
OracleConnection con_cloned = (OracleConnection) con.Clone();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

Close

This method closes the connection to the database.

Declaration

```
// C#
public void Close();
```

Implements

`IDbConnection`

Remarks

Performs the following:

- Rolls back any pending transactions.
- Places the connection to the connection pool if connection pooling is enabled. Even if connection pooling is enabled, the connection can be closed if it exceeds the connection lifetime specified in the connection string. If connection pooling is disabled, the connection is closed.
- Closes the connection to the database.

The connection can be reopened using `Open()`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

CreateCommand

This method creates and returns an `OracleCommand` object associated with the `OracleConnection` object.

Declaration

```
// C#
public OracleCommand CreateCommand();
```

Return Value

The `OracleCommand` object.

Implements

`IDbConnection`

Example

```
// C#
//  Uses connection to create and return an OracleCommand object.

...
string ConStr = "User Id=myschema;Password=mypassword;" +
      "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

OracleCommand cmd = Con.CreateCommand();

cmd.CommandText = "insert into mytable values (99, 'foo')";
cmd.CommandType = CommandType.Text;
cmd.ExecuteNonQuery();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

GetSessionInfo

`GetSessionInfo` returns or refreshes an `OracleGlobalization` object that represents the globalization settings of the session.

Overload List:

- GetSessionInfo()

  This method returns a new instance of the `OracleGlobalization` object that represents the globalization settings of the session.

- GetSessionInfo(OracleGlobalization)

  This method refreshes the provided `OracleGlobalization` object with the globalization settings of the session.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

GetSessionInfo()

This method returns a new instance of the `OracleGlobalization` object that represents the globalization settings of the session.

Declaration

```
// C#
public OracleGlobalization GetSessionInfo();
```

Return Value

The newly created `OracleGlobalization` object.

Example

```
// C#
//  Retrieves the session globalization info and prints the language name.
//  Then sets new territory, language, and timestamp format into the session
//  globalization info in the connection object.

...
string ConStr = "User Id=myschema;Password=mypassword;" +
  "Data Source=oracle;";
```

```
OracleConnection con = new OracleConnection(ConStr);
con.Open();

...
//Get session info from connection object
OracleGlobalization ogi = con.GetSessionInfo();

//Print the language name
Console.WriteLine(ogi.Language);

//Update session info
oraGlob.Territory = "JAPAN";
ogi.Language = "JAPANESE";
ogi.TimeStampFormat = "HH.MI.SSXFF AM YYYY-MM-DD";

//Set session info into connection object
con.SetSessionInfo(ogi);
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

GetSessionInfo(OracleGlobalization)

This method refreshes the provided `OracleGlobalization` object with the globalization settings of the session.

Declaration

```
// C#
public void GetSessionInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

  The `OracleGlobalization` object to be updated.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

Open

This method opens a connection to an Oracle database.

Declaration

```
// C#
public void Open();
```

Implements

`IDbConnection`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The connection is already opened or the connection string is null or empty.

Remarks

The connection is obtained from the pool if connection pooling is enabled. Otherwise, a new connection is established.

It is possible that the pool does not contain any unused connections when the `Open()` method is invoked. In this case, a new connection is established.

If no connections are available within the specified connection timeout value, when the `Max Pool Size` is reached, an `OracleException` is thrown.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

OpenWithNewPassword

This method opens a new connection with the new password.

Declaration

```
// C#
public void OpenWithNewPassword(string newPassword);
```

Parameters

- *newPassword*

  A string that contains the new password.

Remarks

This method uses the `ConnectionString` property settings to establish a new connection. The old password must be provided in the connection string as the `Password` attribute value.

This method can only be called on an `OracleConnection` in the *closed* state.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members
- "Password Expiration" on page 3-6

SetSessionInfo

This method alters the session's globalization settings with all the property values specified in the provided `OracleGlobalization` object.

Declaration

```
// C#
public void SetSessionInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

An `OracleGlobalization` object.

Remarks

Calling this method is equivalent to calling an `ALTER SESSION SQL` on the session.

Example

```
// C#

//  Retrieves the session globalization info and prints the language name.
//  Then sets new territory, language, and timestamp format into the session
//  globalization info in the connection object.

...
string ConStr = "User Id=myschema;Password=mypassword;" +
     "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Create an OracleGlobalization object
OracleGlobalization ogi;

//Get session info using the second overloaded method
con.GetSessionInfo(ogi);

//Print the language name
Console.WriteLine(ogi.Language);

//Update session globalization info
oraGlob.Territory = "JAPAN";
ogi.Language = "JAPANESE";
ogi.TimeStampFormat = "HH.MI.SSXFF AM YYYY-MM-DD";

//Set session globalization info into connection object
con.SetSessionInfo(ogi);
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

OracleConnection Events

`OracleConnection` events are listed in Table 4–29.

Table 4–29 OracleConnection Events

Event Name	Description
`Disposed`	Inherited from `Component`
Failover	An event that is triggered when an Oracle failover occurs
InfoMessage	An event that is triggered for any message or warning sent by the database
StateChange	An event that is triggered when the connection state changes

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

Failover

This event is triggered when an Oracle failover occurs.

Declaration

```
// C#
public event OracleFailoverEventHandler Failover;
```

Event Data

The event handler receives an `OracleFailoverEventArgs` object which exposes the following properties containing information about the event.

- `FailoverType`

  Indicates the type of the failover.

- `FailoverEvent`

  Indicates the state of the failover.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members
- "OracleFailoverEventArgs Properties" on page 4-207

InfoMessage

This event is triggered for any message or warning sent by the database.

Declaration

```
// C#
public event OracleInfoMessageEventHandler InfoMessage;
```

Event Data

The event handler receives an `OracleInfoMessageEventArgs` object which exposes the following properties containing information about the event.

- `Errors`

  The collection of errors generated by the data source.

- `Message`

  The error text generated by the data source.

- `Source`

  The name of the object that generated the error.

Remarks

In order to respond to warnings and messages from the database, the client should create an `OracleInfoMessageEventHandler` delegate to listen to this event.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members
- "OracleInfoMessageEventArgs Properties" on page 4-240

StateChange

This event is triggered when the connection state changes.

Declaration

```
// C#
public event StateChangeEventHandler StateChange;
```

Event Data

The event handler receives a `StateChangeEventArgs` object which exposes the following properties containing information about the event.

- `CurrentState`

  The new state of the connection.

- `OriginalState`

  The original state of the connection.

Remarks

The `StateChange` event is raised after a connection changes state, whenever an explicit call is made to `Open`, `Close` or `Dispose`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members

OracleConnection Event Delegates

`OracleConnection` event delegates are listed in Table 4–30.

Table 4–30 OracleConnection Event Delegates

Event Delegate Name	Description
OracleFailoverEventHandler	An event delegate that handles the `Failover` event
OracleInfoMessageEventHandler	An event delegate that handles the `InfoMessage` event
StateChangeEventHandler	An event delegate that handles the `StateChange` event

OracleFailoverEventHandler

This event delegate handles the `Failover` event.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members
- "OracleTransaction Class" on page 4-323

OracleInfoMessageEventHandler

This event delegate handles the `InfoMessage` event.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members
- "OracleInfoMessageEventHandler Delegate" on page 4-243

StateChangeEventHandler

This event delegate handles the `StateChange` event.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleConnection Class
- OracleConnection Members
- Microsoft ADO.NET documentation for a description of `StateChangeEventHandler`

OracleDataAdapter Class

An OracleDataAdapter object represents a data provider object that populates the DataSet and updates changes in the DataSet to the Oracle database.

Class Inheritance

```
Object
  MarshalByRefObject
    Component
      DataAdapter
        DbDataAdapter
          OracleDataAdapter
```

Declaration

```
// C#
public sealed class OracleDataAdapter : DbDataAdapter, IDbDataAdapter
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

The OracleDataAdapter examples in this section are based on the EMPINFO table which is defined as follows:

```
  CREATE TABLE empInfo (
    empno NUMBER(4) PRIMARY KEY,
    empName VARCHAR2(20) NOT NULL,
    hiredate DATE,
    salary NUMBER(7,2),
    jobDescription Clob,
    byteCodes BLOB
);
```

The EMPINFO table has the following values:

```
EMPNO    EMPNAME    HIREDATE    SALARY     JOBDESCRIPTION  BYTECODES
                                                           (Hex Values)
```

```
=====    =======    ========    ======     ==============  ============

    1    KING       01-MAY-81   12345.67   SOFTWARE ENGR   {0x12, 0x34}
    2    SCOTT      01-SEP-75   34567.89   MANAGER         {0x56, 0x78}
    3    BLAKE      01-OCT-90   9999.12    TRANSPORT       {0x23, 0x45}
    4    SMITH      NULL        NULL       NULL             NULL
```

The following example uses the `OracleDataAdapter` and the dataset to update the `EMPINFO` table:

```
// C#
public static void AdapterUpdate(string connStr)
{
  string cmdStr = "SELECT EMPNO, EMPNAME, SALARY FROM EMPINFO";

  //create the adapter with the selectCommand txt and the
  //connection string
  OracleDataAdapter adapter = new OracleDataAdapter(cmdStr, connStr);

  //get the connection from the adapter
  OracleConnection connection = adapter.SelectCommand.Connection;

  //create the UpdateCommand object for updating the EMPINFO table
  //from the dataset
  adapter.UpdateCommand = new OracleCommand("UPDATE EMPINFO SET SALARY = "+
    " :iSALARY where EMPNO = :iEMPNO", connection);
  adapter.UpdateCommand.Parameters.Add(":iSALARY", OracleDbType.Double,
    0, "SALARY");
  adapter.UpdateCommand.Parameters.Add(":iEMPNO", OracleDbType.Int16,
        0, "EMPNO");

  //Create and fill the DataSet using the EMPINFO
  DataSet dataset = new DataSet();
  adapter.Fill(dataset, "EMPINFO");

  //Get the EMPINFO table from the dataset
  DataTable table = dataset.Tables["EMPINFO"];

  //Get the first row from the EMPINFO table
  DataRow row0 = table.Rows[0];

  //update the salary in the first row
  row0["SALARY"] = 99999.99;

  //Now update the EMPINFO using the adapter, the salary
```

```
  //of 'KING' is changed to 99999.99
  adapter.Update(dataset, "EMPINFO");

}
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Members
- OracleDataAdapter Constructors
- OracleDataAdapter Static Methods
- OracleDataAdapter Properties
- OracleDataAdapter Public Methods
- OracleDataAdapter Events
- OracleDataAdapter Event Delegates

OracleDataAdapter Members

`OracleDataAdapter` members are listed in the following tables:

OracleDataAdapter Constructors

`OracleDataAdapter` constructors are listed in Table 4–31.

Table 4–31 OracleDataAdapter Constructors

Constructor	Description
OracleDataAdapter Constructors	Instantiates a new instance of `OracleDataAdapter` class (Overloaded)

OracleDataAdapter Static Methods

`OracleDataAdapter` static methods are listed in Table 4–32.

Table 4–32 OracleDataAdapter Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleDataAdapter Properties

`OracleDataAdapter` properties are listed in Table 4–33.

Table 4–33 OracleDataAdapter Properties

Name	Description
`AcceptChangesDuringFill`	Inherited from `DataAdapter`
`Container`	Inherited from `Component`
`ContinueUpdateOnError`	Inherited from `DataAdapter`
DeleteCommand	A SQL statement or stored procedure to delete rows from an Oracle database
InsertCommand	A SQL statement or stored procedure to insert new rows into an Oracle database
`MissingMappingAction`	Inherited from `DataAdapter`
`MissingSchemaAction`	Inherited from `DataAdapter`
Requery	Determines whether the `SelectCommand` is reexecuted on the next call to `Fill`
SafeMapping	Creates a mapping between column names in the result set to .NET types, to preserve the data
SelectCommand	A SQL statement or stored procedure that returns a single or multiple result set
`Site`	Inherited from `Component`
`TableMappings`	Inherited from `DataAdapter`
UpdateCommand	A SQL statement or stored procedure to update rows from the `DataSet` to an Oracle database

OracleDataAdapter Public Methods

`OracleDataAdapter` public methods are listed in Table 4–34.

Table 4–34 OracleDataAdapter Public Methods

Public Method	Description
CreateObjRef	Inherited from MarshalByRefObject
Dispose	Inherited from Component
Equals	Inherited from Object (Overloaded)
Fill	Adds or refreshes rows in the DataSet to match the data in the Oracle database (Overloaded)
FillSchema	Inherited from DbDataAdapter
GetFillParameters	Inherited from DbDataAdapter
GetHashCode	Inherited from Object
GetLifetimeService	Inherited from MarshalByRefObject
GetType	Inherited from Object
InitializeLifetimeService	Inherited from MarshalByRefObject
ToString	Inherited from Object
Update	Inherited from DbDataAdapter

OracleDataAdapter Events

OracleDataAdapter events are listed in Table 4–35.

Table 4–35 OracleDataAdapter Events

Event Delegate Name	Description
EventHandler	Inherited from Component
FillErrorEventHandler	Inherited from DbDataAdapter
OracleRowUpdatedEventHandler	Event Delegate for the RowUpdated Event
OracleRowUpdatingEventHandler	Event Delegate for the RowUpdating Event

OracleDataAdapter Event Delegates

OracleDataAdapter event delegates are listed in Table 4–36.

Table 4–36 OracleDataAdapter Event Delegates

Event Delegate Name	Description
`EventHandler`	Inherited from `Component`
`FillErrorEventHandler`	Inherited from `DbDataAdapter`
OracleRowUpdatedEventHandler	Event Delegate for the `RowUpdated` Event
OracleRowUpdatingEventHandler	Event Delegate for the `RowUpdating` Event

OracleDataAdapter Constructors

`OracleDataAdapter` constructors create new instances of an `OracleDataAdapter` class.

Overload List:

- OracleDataAdapter()

  This constructor creates an instance of an `OracleDataAdapter` class.

- OracleDataAdapter(OracleCommand)

  This constructor creates an instance of an `OracleDataAdapter` class with the provided `OracleCommand` as the `SelectCommand`.

- OracleDataAdapter(string, OracleConnection)

  This constructor creates an instance of an `OracleDataAdapter` class with the provided `OracleConnection` object and the command text for the `SelectCommand`.

- OracleDataAdapter(string, string)

  This constructor creates an instance of an `OracleDataAdapter` class with the provided connection string and the command text for the `SelectCommand`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter()

This constructor creates an instance of an `OracleDataAdapter` class with no arguments.

Declaration

```
// C#
public OracleDataAdapter();
```

Remarks

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter(OracleCommand)

This constructor creates an instance of an `OracleDataAdapter` class with the provided `OracleCommand` as the `SelectCommand`.

Declaration

```
// C#
public OracleDataAdapter(OracleCommand selectCommand);
```

Parameters

- *`selectCommand`*

  The `OracleCommand` that is to be set as the `SelectCommand` property.

Remarks

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter(string, OracleConnection)

This constructor creates an instance of an `OracleDataAdapter` class with the provided `OracleConnection` object and the command text for the `SelectCommand`.

Declaration

```
// C#
public OracleDataAdapter(string selectCommandText, OracleConnection
selectConnection);
```

Parameters

- *`selectCommandText`*

  The string that is set as the `CommandText` of the `SelectCommand` property of the `OracleDataAdapter`.

- *`selectConnection`*

  The `OracleConnection` to connect to the Oracle database.

Remarks

The `OracleDataAdapter` opens and closes the connection, if it is not already open. If the connection is open, it must be explicitly closed.

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter(string, string)

This constructor creates an instance of an `OracleDataAdapter` class with the provided connection string and the command text for the `SelectCommand`.

Declaration

```
// C#
public OracleDataAdapter(string selectCommandText, string
selectConnectionString);
```

Parameters

- *`selectCommandText`*

  The string that is set as the `CommandText` of the `SelectCommand` property of the `OracleDataAdapter`.

- *`selectConnectionString`*

  The connection string.

Remarks

Initial values are set for the following `OracleDataAdapter` properties as indicated:

- `MissingMappingAction = MissingMappingAction.Passthrough`
- `MissingSchemaAction = MissingSchemaAction.Add`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter Static Methods

`OracleDataAdapter` static methods are listed in Table 4–37.

Table 4–37 OracleDataAdapter Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter Properties

`OracleDataAdapter` properties are listed in Table 4–38.

Table 4–38 OracleDataAdapter Properties

Name	Description
`AcceptChangesDuringFill`	Inherited from `DataAdapter`
`Container`	Inherited from `Component`
`ContinueUpdateOnError`	Inherited from `DataAdapter`
DeleteCommand	A SQL statement or stored procedure to delete rows from an Oracle database
InsertCommand	A SQL statement or stored procedure to insert new rows into an Oracle database
`MissingMappingAction`	Inherited from `DataAdapter`
`MissingSchemaAction`	Inherited from `DataAdapter`
Requery	Determines whether the `SelectCommand` is reexecuted on the next call to `Fill`
SafeMapping	Creates a mapping between column names in the result set to .NET types, to preserve the data
SelectCommand	A SQL statement or stored procedure that returns a single or multiple result set
`Site`	Inherited from `Component`
`TableMappings`	Inherited from `DataAdapter`
UpdateCommand	A SQL statement or stored procedure to update rows from the `DataSet` to an Oracle database

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

DeleteCommand

This property is a SQL statement or stored procedure to delete rows from an Oracle database.

Declaration

```
// C#
public OracleCommand DeleteCommand {get; set;}
```

Property Value

An `OracleCommand` used during the `Update` call to delete rows from tables in the Oracle database, corresponding to the deleted rows in the `DataSet`.

Remarks

Default = `null`

If there is primary key information in the `DataSet`, the `DeleteCommand` can be automatically generated using the `OracleCommandBuilder`, if no command is provided for this.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

InsertCommand

This property is a SQL statement or stored procedure to insert new rows into an Oracle database.

Declaration

```
// C#
public OracleCommand InsertCommand {get; set;}
```

Property Value

An `OracleCommand` used during the `Update` call to insert rows into a table, corresponding to the inserted rows in the `DataSet`.

Remarks

Default = `null`

If there is primary key information in the `DataSet`, the `InsertCommand` can be automatically generated using the `OracleCommandBuilder`, if no command is provided for this property.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

Requery

This property determines whether the `SelectCommand` is reexecuted on the next call to `Fill`.

Declaration

```
// C#
public Boolean Requery {get; set;}
```

Property Value

Returns `true` if the `SelectCommand` is reexecuted on the next call to `Fill`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-49

SafeMapping

This property creates a mapping between column names in the result set to .NET types that represent column values in the `DataSet`, to preserve the data.

Declaration

```
// C#
public Hashtable SafeMapping {get; set;}
```

Property Value

A hashtable.

Remarks

Default = `null`

The `SafeMapping` property is used, when necessary, to preserve data in the following types:

- `DATE`
- `TimeStamp` (refers to all `TimeStamp` objects)
- `INTERVAL DAY TO SECOND`
- `NUMBER`

Example

```
// C#
// The following example shows how to use the SafeMapping property to fill the
// dataset
public static void UseSafeMapping(
string connStr)
  {

    //In this SELECT statement, EMPNO, HIREDATE and SALARY must be
    //preserved using safe type mapping.
    string cmdStr = "SELECT EMPNO, EMPNAME, HIREDATE, SALARY FROM EMPINFO";

    //create the adapter with the selectCommand txt and the connection string
    OracleDataAdapter adapter = new OracleDataAdapter(cmdStr, connStr);

    //get the connection from the adapter
    OracleConnection connection = adapter.SelectCommand.Connection;

     //create the safe type mapping for the adapter
    //which can safely map column data to byte arrays, where
    // applicable. By executing the following statement, EMPNO, HIREDATE AND
    //SALARY columns will be mapped to byte[]
    adapter.SafeMapping.Add("*", typeof(byte[]));
```

```
    //Map HIREDATE to a string
    //If the column name in the EMPINFO table is case-sensitive,
    //the safe type mapping column name must be case-sensitive.
    adapter.SafeMapping.Add("HIREDATE", typeof(string));

    //Map EMPNO to a string
    //If the column name in the EMPINFO table is case-sensitive,
    //the safe type mapping column name must also be case-sensitive.
    adapter.SafeMapping.Add("EMPNO", typeof(string));

    //Create and fill the DataSet using the EMPINFO
    DataSet dataset = new DataSet();

    adapter.Fill(dataset, "EMPINFO");

    //Get the EMPINFO table from the dataset
    DataTable table = dataset.Tables["EMPINFO"];

    //Get the first row from the EMPINFO table
    DataRow row0 = table.Rows[0];

    //Print out the row info
    Console.WriteLine("EMPNO Column: type = " + row0["EMPNO"].GetType() +
        "; value = " + row0["EMPNO"]);
    Console.WriteLine("EMPNAME Column: type = " + row0["EMPNAME"].GetType() +
        "; value = " + row0["EMPNAME"]);
    Console.WriteLine("HIREDATE Column: type = " + row0["HIREDATE"].GetType()+
        "; value = " + row0["HIREDATE"]);
    Console.WriteLine("SALARY Column: type = " + row0["SALARY"].GetType() +
        "; value = " + row0["SALARY"]);

}
```

Output:

```
EMPNO Column: type = System.String; value = 1
EMPNAME Column: type = System.String; value = KING
HIREDATE Column: type = System.String; value = 01-MAY-81
SALARY Column: type = System.Byte[]; value = System.Byte[]
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Safe Type Mapping" on page 3-46

SelectCommand

This property is a SQL statement or stored procedure that returns single or multiple result sets.

Declaration

```
// C#
public OracleCommand SelectCommand {get; set;}
```

Property Value

An `OracleCommand` used during the `Fill` call to populate the selected rows to the `DataSet`.

Remarks

Default = `null`

If the `SelectCommand` does not return any rows, no tables are added to the dataset and no exception is raised.

If the `SELECT` statement selects from a `VIEW`, no key information is retrieved when a `FillSchema()` or a `Fill()` with `MissingSchemaAction.AddWithKey` is invoked.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-49

UpdateCommand

This property is a SQL statement or stored procedure to update rows from the `DataSet` to an Oracle database.

Declaration

```
// C#
public OracleCommand UpdateCommand {get; set;}
```

Property Value

An `OracleCommand` used during the `Update` call to update rows in the Oracle database, corresponding to the updated rows in the `DataSet`.

Remarks

Default = `null`

If there is primary key information in the `DataSet`, the `UpdateCommand` can be automatically generated using the `OracleCommandBuilder`, if no command is provided for this property.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-49

OracleDataAdapter Public Methods

`OracleDataAdapter` public methods are listed in Table 4–39.

Table 4–39 OracleDataAdapter Public Methods

Public Method	Description
`CreateObjRef`	Inherited from `MarshalByRefObject`
`Dispose`	Inherited from `Component`
`Equals`	Inherited from `Object` (Overloaded)
Fill	Adds or refreshes rows in the `DataSet` to match the data in the Oracle database (Overloaded)
`FillSchema`	Inherited from `DbDataAdapter`
`GetFillParameters`	Inherited from `DbDataAdapter`
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`

Table 4–39 OracleDataAdapter Public Methods (Cont.)

Public Method	Description
GetType	Inherited from Object
InitializeLifetimeService	Inherited from MarshalByRefObject
ToString	Inherited from Object
Update	Inherited from DbDataAdapter

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

Fill

Fill populates or refreshes the specified DataTable or DataSet.

Overload List:

- Fill(DataTable, OracleRefCursor)

  This method adds or refreshes rows in the specified DataTable to match those in the provided OracleRefCursor object.

- Fill(DataSet, OracleRefCursor)

  This method adds or refreshes rows in the DataSet to match those in the provided OracleRefCursor object.

- Fill(DataSet, string, OracleRefCursor)

  This method adds or refreshes rows in the specified source table of the DataSet to match those in the provided OracleRefCursor object.

- Fill(DataSet, int, int, string, OracleRefCursor)

  This method adds or refreshes rows in a specified range in the DataSet to match rows in the provided OracleRefCursor object.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

Fill(DataTable, OracleRefCursor)

This method adds or refreshes rows in the specified `DataTable` to match those in the provided `OracleRefCursor` object.

Declaration

```
// C#
public int Fill(DataTable dataTable, OracleRefCursor refCursor);
```

Parameters

- *`dataTable`*

  The `DataTable` object being populated.

- *`refCursor`*

  The `OracleRefCursor` that rows are being retrieved from.

Return Value

The number of rows added to or refreshed in the `DataTable`.

Exceptions

`ArgumentNullException` - The *`dataTable`* or *`refCursor`* parameter is null.

`InvalidOperationException` - The `OracleRefCursor` is already being used to fetch data.

`NotSupportedException` - The `SafeMapping` type is not supported.

Remarks

No schema or key information is provided, even if the `Fill` method is called with `MissingSchemaAction` set to `MissingSchemaAction.AddWithKey`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-49

Fill(DataSet, OracleRefCursor)

This method adds or refreshes rows in the `DataSet` to match those in the provided `OracleRefCursor` object.

Declaration

```
// C#
public int Fill(DataSet dataSet, OracleRefCursor refCursor);
```

Parameters

- *`dataSet`*

  The `DataSet` object being populated.

- *`refCursor`*

  The `OracleRefCursor` that rows are being retrieved from.

Return Value

Returns the number of rows added or refreshed in the `DataSet`.

Exceptions

`ArgumentNullException` - The *`dataSet`* or *`refCursor`* parameter is null.

`InvalidOperationException` - The `OracleRefCursor` is already being used to fetch data.

`InvalidOperationException` - The `OracleRefCursor` is ready to fetch data.

`NotSupportedException` - The `SafeMapping` type is not supported.

Remarks

If there is no `DataTable` to refresh, a new `DataTable` named `Table` is created and populated using the provided `OracleRefCursor` object.

No schema or key information is provided, even if the `Fill` method is called with `MissingSchemaAction` set to `MissingSchemaAction.AddWithKey`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-49

Fill(DataSet, string, OracleRefCursor)

This method adds or refreshes rows in the specified source table of the `DataSet` to match those in the provided `OracleRefCursor` object.

Declaration

```
// C#
public int Fill(DataSet dataSet, string srcTable, OracleRefCursor refCursor);
```

Parameters

- *`dataSet`*

  The `DataSet` object being populated.

- *`srcTable`*

  The name of the source table used in the table mapping.

- *`refCursor`*

  The `OracleRefCursor` that rows are being retrieved from.

Return Value

Returns the number of rows added or refreshed into the `DataSet`.

Exceptions

`ArgumentNullException` - The *`dataSet`* or *`refCursor`* parameter is null.

`InvalidOperationException` - The `OracleRefCursor` is already being used to fetch data or the source table name is invalid.

`NotSupportedException` - The `SafeMapping` type is not supported.

Remarks

No schema or key information is provided, even if the `Fill` method is called with `MissingSchemaAction` set to `MissingSchemaAction.AddWithKey`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-49

Fill(DataSet, int, int, string, OracleRefCursor)

This method adds or refreshes rows in a specified range in the `DataSet` to match rows in the provided `OracleRefCursor` object.

Declaration

```
// C#
public int Fill(DataSet dataSet, int startRecord, int maxRecords,
  string srcTable, OracleRefCursor refCursor);
```

Parameters

- *dataSet*

  The `DataSet` object being populated.

- *startRecord*

  The record number to start with.

- *maxRecords*

  The maximum number of records to obtain.

- *srcTable*

  The name of the source table used in the table mapping.

- *refCursor*

  The `OracleRefCursor` that rows are being retrieved from.

Return Value

This method returns the number of rows added or refreshed in the `DataSet`. This does not include rows affected by statements that do not return rows.

Exceptions

`ArgumentNullException` - The *`dataSet`* or *`refCursor`* parameter is null.

`InvalidOperationException` - The `OracleRefCursor` is already being used to fetch data or the source table name is invalid.

`NotSupportedException` - The `SafeMapping` type is not supported.

Remarks

No schema or key information is provided, even if the `Fill` method is called with `MissingSchemaAction` set to `MissingSchemaAction.AddWithKey`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-49

OracleDataAdapter Events

`OracleDataAdapter` events are listed in Table 4–40.

Table 4–40 OracleDataAdapter Events

Event Name	Description
`Disposed`	Inherited from `Component`
`FillError`	Inherited from `DbDataAdapter`
RowUpdated	This event is raised when row(s) have been updated by the `Update()` method
RowUpdating	This event is raised when row data are about to be updated to the database

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

RowUpdated

This event is raised when row(s) have been updated by the `Update()` method.

Declaration

```
// C#
public event OracleRowUpdatedEventHandler RowUpdated;
```

Event Data

The event handler receives an `OracleRowUpdatedEventArgs` object which exposes the following properties containing information about the event.

- `Command`

  The `OracleCommand` executed during the `Update`.

- `Errors` (inherited from `RowUpdatedEventArgs`)

  The exception, if any, is generated during the `Update`.

- `RecordsAffected` (inherited from `RowUpdatedEventArgs`)

  The number of rows modified, inserted, or deleted by the execution of the `Command`.

- `Row` (inherited from `RowUpdatedEventArgs`)

  The `DataRow` sent for `Update`.

- `StatementType` (inherited from `RowUpdatedEventArgs`)

  The type of SQL statement executed.

- `Status` (inherited from `RowUpdatedEventArgs`)

  The `UpdateStatus` of the `Command`.

- `TableMapping` (inherited from `RowUpdatedEventArgs`)

  The `DataTableMapping` used during the `Update`.

Example

The following example shows how to use the `RowUpdating` and `RowUpdated` events.

```
// C#
// create the event handler for RowUpdating event

protected static void OnRowUpdating(object sender, OracleRowUpdatingEventArgs e)
```

```
{
  Console.WriteLine("Row updating.....");
  Console.WriteLine("Event arguments:");
  Console.WriteLine("Command Text: " + e.Command.CommandText);
  Console.WriteLine("Command Type: " + e.StatementType);
  Console.WriteLine("Status: " + e.Status);
}

// create the event handler for RowUpdated event
protected static void OnRowUpdated(object sender, OracleRowUpdatedEventArgs e)
{
  Console.WriteLine("Row updated.....");
  Console.WriteLine("Event arguments:");
  Console.WriteLine("Command Text: " + e.Command.CommandText);
  Console.WriteLine("Command Type: " + e.StatementType);
  Console.WriteLine("Status: " + e.Status);
}

public static void AdapterEvents(string connStr)
{
  string cmdStr = "SELECT EMPNO, EMPNAME, SALARY FROM EMPINFO";

  //create the adapter with the selectCommand txt and the
  //connection string
  OracleDataAdapter adapter = new OracleDataAdapter(cmdStr, connStr);

  //get the connection from the adapter
  OracleConnection connection = adapter.SelectCommand.Connection;

  //create the UpdateCommand object for updating the EMPINFO table
  //from the dataset
  adapter.UpdateCommand = new OracleCommand("UPDATE EMPINFO SET SALARY = "+
      " :iSALARY where EMPNO = :iEMPNO", connection);
  adapter.UpdateCommand.Parameters.Add(":iSALARY", OracleDbType.Double,
      0, "SALARY");
  adapter.UpdateCommand.Parameters.Add(":iEMPNO", OracleDbType.Int16,
      0, "EMPNO");

  //Create and fill the DataSet using the EMPINFO
  DataSet dataset = new DataSet();
  adapter.Fill(dataset, "EMPINFO");

  //Get the EMPINFO table from the dataset
  DataTable table = dataset.Tables["EMPINFO"];
```

```
  //Get the first row from the EMPINFO table
  DataRow row0 = table.Rows[0];

  //update the salary in the first row
  row0["SALARY"] = 99999.99;

  //set the event handlers for the RowUpdated and the RowUpdating event
  //the OnRowUpdating() method will be triggered before the update, and
  //the OnRowUpdated() method will be triggered after the update
  adapter.RowUpdating += new OracleRowUpdatingEventHandler(OnRowUpdating);
  adapter.RowUpdated += new OracleRowUpdatedEventHandler(OnRowUpdated);

  //Now update the EMPINFO using the adapter, the salary
  //of 'KING' is changed to 99999.99
  //The OnRowUpdating() and the OnRowUpdated() methods will be triggered
  adapter.Update(dataset, "EMPINFO");
}
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

RowUpdating

This event is raised when row data are about to be updated to the database.

Declaration

```
// C#
public event OracleRowUpdatingEventHandler RowUpdating;
```

Event Data

The event handler receives an `OracleRowUpdatingEventArgs` object which exposes the following properties containing information about the event.

- `Command`

  The `OracleCommand` executed during the `Update`.

- `Errors` (inherited from `RowUpdatingEventArgs`)

  The exception, if any, is generated during the `Update`.

- `Row` (inherited from `RowUpdatingEventArgs`)

  The `DataRow` sent for `Update`.

- `StatementType` (inherited from `RowUpdatingEventArgs`)

  The type of SQL statement executed.

- `Status` (inherited from `RowUpdatingEventArgs`)

  The `UpdateStatus` of the `Command`.

- `TableMapping` (inherited from `RowUpdatingEventArgs`)

  The `DataTableMapping` used during the `Update`.

Example

The example for the `RowUpdated` event also shows how to use the `RowUpdating` event. See `RowUpdated` event "Example" on page 4-108.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter Event Delegates

`OracleDataAdapter` event delegates are listed in Table 4–41.

Table 4–41 OracleDataAdapter Event Delegates

Event Delegate Name	Description
`EventHandler`	Inherited from `Component`
`FillErrorEventHandler`	Inherited from `DbDataAdapter`
OracleRowUpdatedEventHandler	Event Delegate for the `RowUpdated` Event
OracleRowUpdatingEventHandler	Event Delegate for the `RowUpdating` Event

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleRowUpdatedEventHandler

This event delegate handles the `RowUpdated` Event.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- OracleRowUpdatedEventHandler Delegate on page 4-309

OracleRowUpdatingEventHandler

This event delegate handles the `RowUpdating` Event.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleRowUpdatingEventArgs Class" on page 4-316

OracleDataReader Class

An `OracleDataReader` object represents a forward-only, read-only, in-memory result set.

Unlike the `DataSet`, the `OracleDataReader` stays connected and fetches one row at a time.

See Also:

- "Obtaining LONG and LONG RAW Data" on page 3-14
- "Obtaining Data From an OracleDataReader" on page 3-11

Class Inheritance

```
Object
  MarshalByRefObject
    OracleDataReader
```

Declaration

```
// C#
public sealed class OracleDataReader : MarshalByRefObject, IEnumerable,
  IDataReader, IDisposable, IDataRecord
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

An `OracleDataReader` instance is constructed by a call to the `ExecuteReader` method of the `OracleCommand` object. The only properties that can be accessed after the `DataReader` is closed or has been disposed, are `IsClosed` and `RecordsAffected`.

Example

The `OracleDataReader` examples in this section are based on the `EMPINFO` table which is defined as follows:

```
CREATE TABLE empInfo (
```

```
    empno NUMBER(4) PRIMARY KEY,
    empName VARCHAR2(20) NOT NULL,
    hiredate DATE,
    salary NUMBER(7,2),
    jobDescription Clob,
    byteCodes BLOB
);
```

The EMPINFO table has the following values:

```
EMPNO    EMPNAME    HIREDATE    SALARY     JOBDESCRIPTION  BYTECODES
                                                           (Hex Values)
=====    =======    ========    ======     ==============  ============

    1    KING       01-MAY-81   12345.67   SOFTWARE ENGR   {0x12, 0x34}
    2    SCOTT      01-SEP-75   34567.89   MANAGER         {0x56, 0x78}
    3    BLAKE      01-OCT-90   9999.12    TRANSPORT       {0x23, 0x45}
    4    SMITH      NULL        NULL       NULL             NULL
```

The following example retrieves the data from the EMPINFO table:

```
//C #
//This method retrieves all the data from EMPINFO table

public void ReadEmpInfo(string connStr)
{
   string cmdStr = "SELECT * FROM EMPINFO";
   OracleConnection connection = new OracleConnection(connStr);
   OracleCommand cmd = new OracleCommand(cmdStr, connection);
   connection.Open();

   OracleDataReader reader = cmd.ExecuteReader();

   //declare the variables to retrieve the data in EmpInfo
   short empNo;
   string empName;
   DateTime hireDate;
   double salary;
   string jobDesc;
   byte[] byteCodes = new byte[10];

   //read the next row until end of row
   while (reader.Read())
     {
       empNo = reader.GetInt16(0);
```

```
  Console.WriteLine("Employee number: " + empNo);

  empName = reader.GetString(1);
  Console.WriteLine("Employee name: " + empName);

  //the following columns can have NULL value, so it
  //is important to call IsDBNull before getting the column data
 if (!reader.IsDBNull(2))
 {
  hireDate = reader.GetDateTime(2);
  Console.WriteLine("Hire date: " + hireDate);
 }
 if (!reader.IsDBNull(3))
 {
  salary = reader.GetDouble(3);
  Console.WriteLine("Salary: " + salary);
 }
 if (!reader.IsDBNull(4))
 {
  jobDesc = reader.GetString(4);
  Console.WriteLine("Job Description: " + jobDesc);
 }
 if (!reader.IsDBNull(5))
 {
  long len = reader.GetBytes(5, 0, byteCodes, 0, 10);
  Console.Write("Byte codes: " );
  for (int i = 0; i < len; i++)
   Console.Write(byteCodes[i].ToString("x"));
  Console.WriteLine();
 }

 Console.WriteLine();
 //done reading one row
 } //Done Reading EMPINFO table

 //Close the reader
 reader.Close();

 // Close the connection
 connection.Close();
}
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Members
- OracleDataReader Static Methods
- OracleDataReader Properties
- OracleDataReader Public Methods
- OracleDataReader SchemaTable

OracleDataReader Members

`OracleDataReader` members are listed in the following tables:

OracleDataReader Static Methods

`OracleDataReader` static methods are listed in Table 4–42.

Table 4–42 OracleDataReader Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleDataReader Properties

`OracleDataReader` properties are listed in Table 4–43.

Table 4–43 OracleDataReader Properties

Property	Description
Depth	Gets a value indicating the depth of nesting for the current row
FetchSize	Specifies the size of `OracleDataReader`'s internal cache
FieldCount	Gets the number of columns in the result set
IsClosed	Indicates whether the data reader is closed

Table 4–43 OracleDataReader Properties (Cont.)

Property	Description
Item	Gets the value of the column (Overloaded)
InitialLOBFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns
RecordsAffected	Gets the number of rows changed, inserted, or deleted by execution of the SQL statement

OracleDataReader Public Methods

`OracleDataReader` public methods are listed in Table 4–44.

Table 4–44 OracleDataReader Public Methods

Public Method	Description
Close	Closes the `OracleDataReader`
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Releases any resources or memory allocated by the object
`Equals`	Inherited from `Object` (Overloaded)
`GetBoolean`	*Not Supported*
GetByte	Returns the byte value of the specified column
GetBytes	Populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column
`GetChar`	*Not Supported*
GetChars	Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column
`GetData`	*Not Supported*
GetDataTypeName	Returns the ODP.NET type name of the specified column
GetDateTime	Returns the `DateTime` value of the specified column

Table 4–44 OracleDataReader Public Methods (Cont.)

Public Method	Description
GetDecimal	Returns the `decimal` value of the specified `NUMBER` column
GetDouble	Returns the `double` value of the specified `NUMBER` column or `BINARY_DOUBLE` column
GetFieldType	Returns the `Type` of the specified column
GetFloat	Returns the `float` value of the specified `NUMBER` column or `BINARY_FLOAT` column
`GetGuid`	*Not Supported*
`GetHashCode`	Inherited from `Object`
GetInt16	Returns the `Int16` value of the specified `NUMBER` column
GetInt32	Returns the `Int32` value of the specified `NUMBER` column
GetInt64	Returns the `Int64` value of the specified `NUMBER` column
`GetLifetimeService`	Inherited by `MarshalByRefObject`
GetName	Returns the name of the specified column
GetOracleBFile	Returns an `OracleBFile` object of the specified `BFILE` column
GetOracleBinary	Returns an `OracleBinary` structure of the specified column
GetOracleBlob	Returns an `OracleBlob` object of the specified `BLOB` column
GetOracleBlobForUpdate	Returns an updatable `OracleBlob` object of the specified `BLOB` column
GetOracleClob	Returns an `OracleClob` object of the specified `CLOB` column
GetOracleClobForUpdate	Returns an updatable `OracleClob` object of the specified `CLOB` column
GetOracleDate	Returns an `OracleDate` structure of the specified `DATE` column
GetOracleDecimal	Returns an `OracleDecimal` structure of the specified `NUMBER` column

Table 4–44 OracleDataReader Public Methods (Cont.)

Public Method	Description
GetOracleIntervalDS	Returns an `OracleIntervalDS` structure of the specified `INTERVAL DAY TO SECOND` column
GetOracleIntervalYM	Returns an `OracleIntervalYM` structure of the specified `INTERVAL YEAR TO MONTH` column
GetOracleString	Returns an `OracleString` structure of the specified column
GetOracleTimeStamp	Returns an `OracleTimeStamp` structure of the Oracle `TimeStamp` column
GetOracleTimeStampLTZ	Returns an `OracleTimeStampLTZ` structure of the specified Oracle `TimeStamp WITH LOCAL TIME ZONE` column
GetOracleTimeStampTZ	Returns an `OracleTimeStampTZ` structure of the specified Oracle `TimeStamp WITH TIME ZONE` column
GetOracleXmlType	Returns an `OracleXmlType` object of the specified `XMLType` column
GetOracleValue	Returns the specified column value as a ODP.NET type
GetOracleValues	Gets all the column values as ODP.NET types
GetOrdinal	Returns the `0`-based ordinal (or index) of the specified column name
GetSchemaTable	Returns a `DataTable` that describes the column metadata of the `OracleDataReader`
GetString	Returns the string value of the specified column
GetTimeSpan	Returns the `TimeSpan` value of the specified `INTERVAL DAY TO SECOND` column
`GetType`	Inherited from `Object` class
GetValue	Returns the column value as a .NET type
GetValues	Gets all the column values as .NET types
GetXmlReader	Returns the contents of an `XMLType` column as an instance of an .NET `XmlTextReader` object
IsDBNull	Indicates whether the column value is null
NextResult	Advances the data reader to the next result set when reading the results

Table 4–44 OracleDataReader Public Methods (Cont.)

Public Method	Description
Read	Reads the next row in the result set
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class

OracleDataReader Static Methods

`OracleDataReader` static methods are listed in Table 4–45.

Table 4–45 OracleDataReader Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

OracleDataReader Properties

`OracleDataReader` properties are listed in Table 4–46.

Table 4–46 OracleDataReader Properties

Property	Description
Depth	Gets a value indicating the depth of nesting for the current row
FetchSize	Specifies the size of `OracleDataReader`'s internal cache
FieldCount	Gets the number of columns in the result set
IsClosed	Indicates whether the data reader is closed
Item	Gets the value of the column (Overloaded)

Table 4–46 OracleDataReader Properties (Cont.)

Property	Description
InitialLOBFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for LOB columns
InitialLONGFetchSize	Specifies the amount that the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns
RecordsAffected	Gets the number of rows changed, inserted, or deleted by execution of the SQL statement

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

Depth

This property gets a value indicating the depth of nesting for the current row.

Declaration

```
// C#
public int Depth {get;}
```

Property Value

The depth of nesting for the current row.

Implements

`IDataReader`

Exceptions

`InvalidOperationException` - The reader is closed.

Remarks

Default = `0`

This property always returns zero because Oracle does not support nesting.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

FetchSize

This property specifies the size of `OracleDataReader`'s internal cache.

Declaration

```
// C#
public long FetchSize {get; set;}
```

Property Value

A `long` that specifies the amount of memory (in bytes) that the `OracleDataReader` uses for its internal cache.

Exceptions

`ArgumentException` - The `FetchSize` value specified is invalid.

Remarks

Default = The `OracleCommand`'s `FetchSize` property value.

The `FetchSize` property is inherited by the `OracleDataReader` that is created by a command execution returning a result set. The `FetchSize` property on the `OracleDataReader` object determines the amount of data fetched into its internal cache for each server round-trip.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- `OracleCommand` "ExecuteReader()" on page 4-33
- `OracleCommand` "RowSize" on page 4-23

FieldCount

This property gets the number of columns in the result set.

Declaration

```
// C#
public int FieldCount {get;}
```

Property Value

The number of columns in the result set if one exists, otherwise `0`.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed.

Remarks

Default = `0`

This property has a value of 0 for queries that do not return result sets.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

IsClosed

This property indicates whether the data reader is closed.

Declaration

```
// C#
public bool IsClosed {get;}
```

Property Value

If the `OracleDataReader` is in a closed state, returns `true`; otherwise, returns `false`.

Implements

`IDataReader`

Remarks

Default = `true`

`IsClosed` and `RecordsAffected` are the only two properties that are accessible after the `OracleDataReader` is closed.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

Item

This property gets the value of the column in .NET datatype.

Overload List:

- Item [index]

  This property gets the .NET `Value` of the column specified by the column index.

- Item [string]

  This property gets the .NET `Value` of the column specified by the column name.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

Item [index]

This property gets the .NET `Value` of the column specified by the column index.

Declaration

```
// C#
public object this[int index] {get;}
```

Parameters

- *index*

The zero-based index of the column.

Property Value

The .NET value of the specified column.

Implements

`IDataRecord`

Remarks

Default = Not Applicable

In C#, this property is the indexer for this class.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

Item [string]

This property gets the .NET `Value` of the column specified by the column name.

Declaration

```
// C#
public object this[string columnName] {get;}
```

Parameters

- *`columnName`*

  The name of the column.

Property Value

The .NET `Value` of the specified column.

Implements

`IDataRecord`

Remarks

Default = Not Applicable

A case-sensitive search is made to locate the specified column by its name. If this fails, then a case-insensitive search is made.

In C#, this property is the indexer for this class.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

InitialLOBFetchSize

This property specifies the amount that the `OracleDataReader` initially fetches for LOB columns.

Declaration

```
// C#
public int InitialLOBFetchSize {get;}
```

Property Value

The size of the chunk to retrieve.

Exceptions

`InvalidOperationException` - The reader is closed.

Remarks

The maximum value supported for `InitialLOBFetchSize` is `32767`. If this property is set to a higher value, the provider resets it to `32767`.

Default is the `OracleCommand.InitialLOBFetchSize`, from which this value is inherited.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "InitialLOBFetchSize" on page 4-20 for further information on `OracleCommand.InitialLOBFetchSize`
- "Obtaining LOB Data" on page 3-15

InitialLONGFetchSize

This property specifies the amount that the `OracleDataReader` initially fetches for `LONG` and `LONG RAW` columns.

Declaration

```
// C#
public long InitialLONGFetchSize {get;}
```

Property Value

The size of the chunk to retrieve. The default is `0`.

Exceptions

`InvalidOperationException` - The reader is closed.

Remarks

The maximum value supported for `InitialLONGFetchSize` is `32767`. If this property is set to a higher value, the provider resets it to `32767`.

Default is `OracleCommand.InitialLONGFetchSize`, from which this value is inherited.

This property is read-only for the `OracleDataReader`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "InitialLONGFetchSize" on page 4-21 for further information on OracleCommand.InitialLONGFetchSize
- "Obtaining LONG and LONG RAW Data" on page 3-14

RecordsAffected

This property gets the number of rows changed, inserted, or deleted by execution of the SQL statement.

Declaration

```
// C#
public int RecordsAffected {get;}
```

Property Value

The number of rows affected by execution of the SQL statement.

Implements

`IDataReader`

Remarks

Default = `0`

The value of `-1` is returned for `SELECT` statements.

`IsClosed` and `RecordsAffected` are the only two properties that are accessible after the `OracleDataReader` is closed.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

OracleDataReader Public Methods

`OracleDataReader` public methods are listed in Table 4–47.

Table 4–47 OracleDataReader Public Methods

Public Method	Description
Close	Closes the `OracleDataReader`
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Releases any resources or memory allocated by the object
`Equals`	Inherited from `Object` (Overloaded)
`GetBoolean`	*Not Supported*
GetByte	Returns the byte value of the specified column
GetBytes	Populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column
`GetChar`	*Not Supported*
GetChars	Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column
`GetData`	*Not Supported*
GetDataTypeName	Returns the ODP.NET type name of the specified column
GetDateTime	Returns the `DateTime` value of the specified column
GetDecimal	Returns the `decimal` value of the specified `NUMBER` column
GetDouble	Returns the `double` value of the specified `NUMBER` column or `BINARY_DOUBLE` column
GetFieldType	Returns the `Type` of the specified column
GetFloat	Returns the `float` value of the specified `NUMBER` column or `BINARY_FLOAT` column
`GetGuid`	*Not Supported*
`GetHashCode`	Inherited from `Object`
GetInt16	Returns the `Int16` value of the specified `NUMBER` column

Table 4–47 OracleDataReader Public Methods (Cont.)

Public Method	Description
GetInt32	Returns the `Int32` value of the specified `NUMBER` column
GetInt64	Returns the `Int64` value of the specified `NUMBER` column
`GetLifetimeService`	Inherited by `MarshalByRefObject`
GetName	Returns the name of the specified column
GetOracleBFile	Returns an `OracleBFile` object of the specified `BFILE` column
GetOracleBinary	Returns an `OracleBinary` structure of the specified column
GetOracleBlob	Returns an `OracleBlob` object of the specified `BLOB` column
GetOracleBlobForUpdate	Returns an updatable `OracleBlob` object of the specified `BLOB` column
GetOracleClob	Returns an `OracleClob` object of the specified `CLOB` column
GetOracleClobForUpdate	Returns an updatable `OracleClob` object of the specified `CLOB` column
GetOracleDate	Returns an `OracleDate` structure of the specified `DATE` column
GetOracleDecimal	Returns an `OracleDecimal` structure of the specified `NUMBER` column
GetOracleIntervalDS	Returns an `OracleIntervalDS` structure of the specified `INTERVAL DAY TO SECOND` column
GetOracleIntervalYM	Returns an `OracleIntervalYM` structure of the specified `INTERVAL YEAR TO MONTH` column
GetOracleString	Returns an `OracleString` structure of the specified column
GetOracleTimeStamp	Returns an `OracleTimeStamp` structure of the Oracle `TimeStamp` column
GetOracleTimeStampLTZ	Returns an `OracleTimeStampLTZ` structure of the specified Oracle `TimeStamp WITH LOCAL TIME ZONE` column

Table 4–47 OracleDataReader Public Methods (Cont.)

Public Method	Description
GetOracleTimeStampTZ	Returns an `OracleTimeStampTZ` structure of the specified Oracle `TimeStamp WITH TIME ZONE` column
GetOracleXmlType	Returns an `OracleXmlType` object of the specified `XMLType` column
GetOracleValue	Returns the specified column value as a ODP.NET type
GetOracleValues	Gets all the column values as ODP.NET types
GetOrdinal	Returns the `0`-based ordinal (or index) of the specified column name
GetSchemaTable	Returns a `DataTable` that describes the column metadata of the `OracleDataReader`
GetString	Returns the string value of the specified column
GetTimeSpan	Returns the `TimeSpan` value of the specified `INTERVAL DAY TO SECOND` column
`GetType`	Inherited from `Object` class
GetValue	Returns the column value as a .NET type
GetValues	Gets all the column values as .NET types
GetXmlReader	Returns the value of an `XMLType` column as an instance of an .NET `XmlTextReader`
IsDBNull	Indicates whether the column value is null
NextResult	Advances the data reader to the next result set when reading the results
Read	Reads the next row in the result set
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

Close

This method closes the `OracleDataReader`.

Declaration

```
// C#
public void Close();
```

Implements

`IDataReader`

Remarks

The `Close` method frees all resources associated with the `OracleDataReader`.

Example

The code example for the `OracleDataReader` class includes the `Close` method. See `OracleDataReader` Overview "Example" on page 4-113.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

The `Dispose` method also closes the `OracleDataReader`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetByte

This method returns the byte value of the specified column.

Declaration

```
// C#
public byte GetByte(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The value of the column as a byte.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetBytes

This method populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column.

Declaration

```
// C#
public long GetBytes(int index, long fieldOffset, byte[] buffer, int
bufferOffset, int length);
```

Parameters

- *index*

  The zero-based column index.

- *fieldOffset*

  The offset within the column from which reading begins (in bytes).

- *buffer*

  The byte array that the data is read into.

- *bufferOffset*

  The offset within the buffer to begin reading data into (in bytes).

- *length*

  The maximum number of bytes to read (in bytes).

Return Value

The number of bytes read.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

This method returns the number of bytes read into the buffer. This may be less than the actual length of the field if the method has been called previously for the same column.

If a null reference is passed for buffer, the length of the field in bytes is returned.

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetChars

This method populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column.

Declaration

```
// C#
public long GetChars(int index, long fieldOffset, char[] buffer, int
bufferOffset, int length);
```

Parameters

- *index*

  The zero based column index.

- *fieldOffset*

  The index within the column from which to begin reading (in characters).

- *buffer*

  The character array that the data is read into.

- *bufferOffset*

  The index within the buffer to begin reading data into (in characters).

- *length*

  The maximum number of characters to read (in characters).

Return Value

The number of characters read.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

This method returns the number of characters read into the buffer. This may be less than the actual length of the field, if the method has been called previously for the same column.

If a null reference is passed for buffer, the length of the field in characters is returned.

IsDBNull should be called to check for NULL values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetDataTypeName

This method returns the ODP.NET type name of the specified column.

Declaration

```
// C#
public string GetDataTypeName(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The name of the ODP.NET type of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed.

`IndexOutOfRangeException` - The column index is invalid.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetDateTime

This method returns the `DateTime` value of the specified column.

Declaration

```
// C#
public DateTime GetDateTime(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `DateTime` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetDecimal

This method returns the decimal value of the specified NUMBER column.

Declaration

```
// C#
public decimal GetDecimal(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The decimal value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetDouble

This method returns the double value of the specified NUMBER column or BINARY_DOUBLE column.

Declaration

```
// C#
public double GetDouble(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The double value of the column.

Implements

IDataRecord

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

Starting with Oracle Database 10*g*, `GetDouble` now supports retrieval of data from `BINARY_DOUBLE` columns.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetFieldType

This method returns the `Type` of the specified column.

Declaration

```
// C#
public Type GetFieldType(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `Type` of the default .NET type of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed.

`IndexOutOfRangeException` - The column index is invalid.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetFloat

This method returns the `float` value of the specified `NUMBER` column or `BINARY_FLOAT` column.

Declaration

```
// C#
public float GetFloat(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `float` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

Starting with Oracle Database 10*g*, `GetFloat` now supports retrieval of data from `BINARY_FLOAT` columns.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetInt16

This method returns the `Int16` value of the specified `NUMBER` column.

> **Note:** `short` is equivalent to `Int16`.

Declaration

```
// C#
public short GetInt16(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `Int16` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetInt32

This method returns the `Int32` value of the specified `NUMBER` column.

> **Note:** `int` is equivalent to `Int32`.

Declaration

```
// C#
public int GetInt32(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `Int32` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetInt64

This method returns the `Int64` value of the specified `NUMBER` column.

> **Note:** `long` is equivalent to `Int64`.

Declaration

```
// C#
public long GetInt64(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `Int64` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetName

This method returns the name of the specified column.

Declaration

```
// C#
public string GetName(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The name of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed.

`IndexOutOfRangeException` - The column index is invalid.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleBFile

This method returns an `OracleBFile` object of the specified `BFILE` column.

Declaration

```
// C#
public OracleBFile GetOracleBFile(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleBFile` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleBinary

This method returns an `OracleBinary` structure of the specified column.

Declaration

```
// C#
public OracleBinary GetOracleBinary(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleBinary` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

GetOracleBinary is used on the following Oracle types:

- BFILE
- BLOB
- LONG RAW
- RAW

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleBlob

This method returns an OracleBlob object of the specified BLOB column.

Declaration

```
// C#
public OracleBlob GetOracleBlob(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The OracleBlob value of the column.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleBlobForUpdate

`GetOracleBlobForUpdate` returns an updatable `OracleBlob` object of the specified `BLOB` column.

Overload List:

- GetOracleBlobForUpdate(int)

  This method returns an updatable `OracleBlob` object of the specified `BLOB` column.

- GetOracleBlobForUpdate(int, int)

  This method returns an updatable `OracleBlob` object of the specified `BLOB` column using a `WAIT` clause.

GetOracleBlobForUpdate(int)

This method returns an updatable `OracleBlob` object of the specified `BLOB` column.

Declaration

```
// C#
public OracleBlob GetOracleBlobForUpdate(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

An updatable OracleBlob object.

Exceptions

InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

When the OracleCommand's ExecuteReader() method is invoked, all the data fetched by the OracleDataReader is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the GetOracleBlobForUpdate() method incurs a server round-trip to obtain a reference to the current BLOB data while also locking the row using the FOR UPDATE clause. This means that the OracleBlob obtained from GetOracleBlob() can have a different value than the OracleBlob obtained from GetOracleBlobForUpdate() since it is not obtained from the original snapshot.

The returned OracleBlob object can be used to safely update the BLOB because the BLOB column has been locked after a call to this method.

Invoking this method internally executes a SELECT..FOR UPDATE statement without a WAIT clause. Therefore, the statement can wait indefinitely until a lock is acquired for that row.

IsDBNull should be called to check for NULL values before calling this method.

Example

The following example gets the OracleBlob object for update from the reader, updates the OracleBlob object, and then commits the transaction.

```
// C#
public static void ReadOracleBlobForUpdate(string connStr)
  {
   //get the job description for empno = 1
   string cmdStr = "SELECT BYTECODES, EMPNO FROM EMPINFO where EMPNO = 1";

   OracleConnection connection = new OracleConnection(connStr);
   OracleCommand cmd = new OracleCommand(cmdStr, connection);
```

```
  connection.Open();

  //Since we are going to update the OracleBlob object, we will
  //have to create a transaction
  OracleTransaction txn = connection.BeginTransaction();

  //get the reader
  OracleDataReader reader = cmd.ExecuteReader();

  //declare the variables to retrieve the data in EmpInfo
  OracleBlob byteCodesBlob;

  //read the first row
  reader.Read();

  if (!reader.IsDBNull(0))
  {
    byteCodesBlob = reader.GetOracleBlobForUpdate(0);

    //Close the reader
    reader.Close();

    //Update the job description Clob object
    byte[] addedBytes = new byte[2] {0, 0};
    byteCodesBlob.Append(addedBytes, 0, addedBytes.Length);

    //Now commit the transaction
    txn.Commit();
   }
   else
     reader.Close();

   // Close the connection
   connection.Close();
}
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-35

GetOracleBlobForUpdate(int, int)

This method returns an updatable `OracleBlob` object of the specified `BLOB` column using a `WAIT` clause.

Declaration

```
// C#
public OracleBlob GetOracleBlobForUpdate(int index, int wait);
```

Parameters

- *index*

  The zero-based column index.

- *wait*

  The number of seconds the method waits to acquire a lock.

Return Value

An updatable `OracleBlob` object.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

When the `OracleCommand`'s `ExecuteReader()` method is invoked, all the data fetched by the `OracleDataReader` is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the `GetOracleBlobForUpdate()` method incurs a server round-trip to obtain a reference to the current `BLOB` data while also locking the row using the `FOR UPDATE` clause. This means that the `OracleBlob` obtained from `GetOracleBlob()` can have a different value than the `OracleBlob` obtained from `GetOracleBlobForUpdate()` since it is not obtained from the original snapshot.

`IsDBNull` should be called to check for `NULL` values before calling this method.

The returned OracleBlob object can be used to safely update the BLOB because the BLOB column has been locked after a call to this method.

Invoking this method internally executes a SELECT..FOR UPDATE statement which locks the row.

Different WAIT clauses are appended to the statement, depending on the *wait* value. If the *wait* value is:

- 0

  "NOWAIT" is appended at the end of a SELECT..FOR UPDATE statement. The statement executes immediately whether the lock is acquired or not. If the lock is not acquired, an exception is thrown.

- *n*

  "WAIT *n*" is appended at the end of a SELECT..FOR UPDATE statement. The statement executes as soon as the lock is acquired. However, if the lock cannot be acquired by *n* seconds, this method call throws an exception.

  The WAIT *n*" feature is only available for Oracle9*i* or later. For any version lower than Oracle9*i*, *n* is implicitly treated as -1 and nothing is appended at the end of a SELECT..FOR UPDATE statement.

- -1

  Nothing is appended at the end of the SELECT..FOR UPDATE. The statement execution waits indefinitely until a lock can be acquired.

Example

The GetOracleBlobForUpdate methods are comparable. See "Example" on page 4-149 for a code example demonstrating usage.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-35

GetOracleClob

This method returns an OracleClob object of the specified CLOB column.

Declaration

```
// C#
public OracleClob GetOracleClob(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleClob` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-35

GetOracleClobForUpdate

`GetOracleClobForUpdate` returns an updatable `OracleClob` object of the specified `CLOB` column.

Overload List:

- GetOracleClobForUpdate(int)

  This method returns an updatable `OracleClob` object of the specified `CLOB` column.

- GetOracleClobForUpdate(int, int)

  This method returns an updatable `OracleClob` object of the specified `CLOB` column using a `WAIT` clause.

GetOracleClobForUpdate(int)

This method returns an updatable `OracleClob` object of the specified `CLOB` column.

Declaration

```
// C#
public OracleClob GetOracleClobForUpdate(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

An updatable `OracleClob`.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

When the `OracleCommand`'s `ExecuteReader()` method is invoked, all the data fetched by the `OracleDataReader` is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the `GetOracleClobForUpdate()` method incurs a server round-trip to obtain a reference to the current `CLOB` data while also locking the row using the `FOR UPDATE` clause. This means that the `OracleClob` obtained from `GetOracleClob()` can have a different value than the `OracleClob` obtained from `GetOracleClobForUpdate()` since it is not obtained from the original snapshot.

The returned OracleClob object can be used to safely update the CLOB because the CLOB column is locked after a call to this method.

Invoking this method internally executes a SELECT..FOR UPDATE statement without a WAIT clause. Therefore, the statement can wait indefinitely until a lock is acquired for that row.

IsDBNull should be called to check for NULL values before calling this method.

Example

The following example gets the OracleClob object for update from the reader, updates the OracleClob object, and then commits the transaction.

```
// C#
public static void ReadOracleClobForUpdate(string connStr)
  {
    //get the job description for empno = 1
    string cmdStr = "SELECT JOBDESCRIPTION, EMPNO FROM EMPINFO where EMPNO = 1";

    OracleConnection connection = new OracleConnection(connStr);
    OracleCommand cmd = new OracleCommand(cmdStr, connection);
    connection.Open();

    //Since we are going to update the OracleClob object, we will
    //have to create a transaction
    OracleTransaction txn = connection.BeginTransaction();

    //get the reader
    OracleDataReader reader = cmd.ExecuteReader();

    //declare the variables to retrieve the data in EmpInfo
    OracleClob jobDescClob;

    //read the first row
    reader.Read();

    if (!reader.IsDBNull(0))
    {
      jobDescClob = reader.GetOracleClobForUpdate(0);

      //Close the reader
      reader.Close();

      //Update the job description Clob object
      char[] jobDesc = "-SALES".ToCharArray();
```

```
    jobDescClob.Append(jobDesc, 0, jobDesc.Length);

    //Now commit the transaction
    txn.Commit();
  }
  else
    reader.Close();

  // Close the connection
  connection.Close();
}
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-35

GetOracleClobForUpdate(int, int)

This method returns an updatable `OracleClob` object of the specified `CLOB` column using a `WAIT` clause.

Declaration

```
// C#
public OracleClob GetOracleClobForUpdate(int index, int wait);
```

Parameters

- *index*

  The zero-based column index.

- *wait*

  The number of seconds the method waits to acquire a lock.

Return Value

An updatable `OracleClob`.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

When the `OracleCommand`'s `ExecuteReader()` method is invoked, all the data fetched by the `OracleDataReader` is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the `GetOracleClobForUpdate()` method incurs a server round-trip to obtain a reference to the current `CLOB` data while also locking the row using the `FOR UPDATE` clause. This means that the `OracleClob` obtained from `GetOracleClob()` can have a different value than the `OracleClob` obtained from `GetOracleClobForUpdate()` since it is not obtained from the original snapshot.

Invoking this method internally executes a `SELECT..FOR UPDATE` statement which locks the row.

The returned `OracleClob` object can be used to safely update the `CLOB` because the `CLOB` column is locked after a call to this method.

Different `WAIT` clauses are appended to the statement, depending on the *`wait`* value. If the *`wait`* value is:

- `0`

  "`NOWAIT`" is appended at the end of a `SELECT..FOR UPDATE` statement. The statement executes immediately whether the lock is acquired or not. If the lock is not acquired, an exception is thrown.

- *`n`*

  "`WAIT` *`n`*" is appended at the end of a `SELECT..FOR UPDATE` statement. The statement executes as soon as the lock is acquired. However, if the lock cannot be acquired by *`n`* seconds, this method call throws an exception.

  The `WAIT` *`n`*" feature is only available for Oracle9*i* or later. For any version lower than Oracle9*i*, *`n`* is implicitly treated as `-1` and nothing is appended at the end of a `SELECT..FOR UPDATE` statement.

- `-1`

Nothing is appended at the end of the `SELECT..FOR UPDATE`. The statement execution waits indefinitely until a lock can be acquired.

`IsDBNull` should be called to check for `NULL` values before calling this method.

Example

The GetOracleClobForUpdate methods are comparable. See "Example" on page 4-155 for a code example demonstrating usage.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-35

GetOracleDate

This method returns an OracleDate structure of the specified DATE column.

Declaration

```
// C#
public OracleDate GetOracleDate(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleDate` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-35

GetOracleDecimal

This method returns an `OracleDecimal` structure of the specified `NUMBER` column.

Declaration

```
// C#
public OracleDecimal GetOracleDecimal(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleDecimal` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleIntervalDS

This method returns an `OracleIntervalDS` structure of the specified `INTERVAL DAY TO SECOND` column.

Declaration

```
// C#
public OracleIntervalDS GetOracleIntervalDS(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleIntervalDS` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleIntervalYM

This method returns an `OracleIntervalYM` structure of the specified `INTERVAL YEAR TO MONTH` column.

Declaration

```
// C#
public OracleIntervalYM GetOracleIntervalYM(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleIntervalYM` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleString

This method returns an `OracleString` structure of the specified column. The string is stored as a Unicode string.

Declaration

```
// C#
```

```
public OracleString GetOracleString(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleString` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

`GetOracleString` is used on the following Oracle column types:

- `CHAR`
- `CLOB`
- `LONG`
- `NCLOB`
- `NCHAR`
- `NVARCHAR2`
- `ROWID`
- `UROWID`
- `VARCHAR2`
- `XMLType`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleTimeStamp

This method returns an `OracleTimeStamp` structure of the Oracle `TimeStamp` column.

Declaration

```
// C#
public OracleTimeStamp GetOracleTimeStamp(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleTimeStamp` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`GetOracleTimeStamp` is used with the Oracle Type `TimeStamp`.

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleTimeStampLTZ

This method returns an `OracleTimeStampLTZ` structure of the specified Oracle `TimeStamp WITH LOCAL TIME ZONE` column.

Declaration

```
// C#
public OracleTimeStampLTZ GetOracleTimeStampLTZ(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleTimeStampLTZ` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`GetOracleTimeStampLTZ` is used with the Oracle Type `TimeStamp` with Local Time Zone columns.

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleTimeStampTZ

This method returns an `OracleTimeStampTZ` structure of the specified Oracle `TimeStamp WITH TIME ZONE` column.

Declaration

```
// C#
public OracleTimeStampTZ GetOracleTimeStampTZ(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleTimeStampTZ` value of the column.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

Used with the Oracle Type `TimeStamp` with Local Time Zone columns

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleXmlType

This method returns an `OracleXmlType` object of the specified `XMLType` column.

Declaration

```
// C#
public OracleXmlType GetOracleXmlType(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `OracleXmlType` value of the column.

Exceptions

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

Requirements

This property can only be used with Oracle9*i* Release 2 (9.2) or higher.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleValue

This method returns the specified column value as an ODP.NET type.

Declaration

```
// C#
public object GetOracleValue(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The value of the column as an ODP.NET type.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetOracleValues

This method gets all the column values as ODP.NET types.

Declaration

```
// C#
public int GetOracleValues(object[] values);
```

Parameters

- *values*

  An array of objects to hold the ODP.NET types as the column values.

Return Value

The number of ODP.NET types in the *values* array.

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

Remarks

This method provides a way to retrieve all column values rather than retrieving each column value individually.

The number of column values retrieved is the minimum of the length of the *values* array and the number of columns in the result set.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-35

GetOrdinal

This method returns the `0`-based ordinal (or index) of the specified column name.

Declaration

```
// C#
public int GetOrdinal(string name);
```

Parameters

- *name*

  The specified column name.

Return Value

The index of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed.

`IndexOutOfRangeException` - The column index is invalid.

Remarks

A case-sensitive search is made to locate the specified column by its name. If this fails, then a case-insensitive search is made.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetSchemaTable

This method returns a `DataTable` that describes the column metadata of the `OracleDataReader`.

Declaration

```
// C#
public DataTable GetSchemaTable();
```

Return Value

A `DataTable` that contains the metadata of the result set.

Implements

`IDataReader`

Exceptions

`InvalidOperationException` - The connection is closed or the reader is closed.

Remarks

`OracleDataReader.GetSchemaTable()` returns the `SchemaTable`.

OracleDataReader SchemaTable

The `OracleDataReader SchemaTable` is a `DataTable` that describes the column metadata of the `OracleDataReader`.

The columns of the SchemaTable are in the order shown.

Table 4–48 OracleDataReader SchemaTable

Name	Name Type	Description
ColumnNameTB	System.String	The name of the column.
ColumnOrdinal	System.Int32	The 0-based ordinal of the column.
ColumnSize	System.Int64	The maximum possible length of a value in the column. ColumnSize value is determined as follows: ■ CHAR and VARCHAR2 types: in bytes - if IsByteSemantic boolean value is true in characters - if IsByteSemantic boolean value is false ■ All other types: in bytes See "IsByteSemantic" on page 4-173 for more information.
NumericPrecision	System.Int16	The maximum precision of the column, if the column is a numeric datatype. This column has valid values for Oracle NUMBER, Oracle INTERVAL YEAR TO MONTH, and Oracle INTERVAL DAY TO SECOND columns. For all other columns, the value is null.
NumericScale	System.Int16	The scale of the column. This column has valid values for Oracle NUMBER, Oracle INTERVAL DAY TO SECOND, and the Oracle TIMESTAMP columns. For all other columns, the value is null.

Table 4–48 OracleDataReader SchemaTable (Cont.)

Name	Name Type	Description
`IsUnique`	`System.Boolean`	Indicates whether the column is unique. `true` if no two rows in the base table can have the same value in this column, where the base table is the table returned in `BaseTableName`. `IsUnique` is guaranteed to be `true` if one of the following applies: ■ the column constitutes a key by itself ■ there is a unique constraint or a unique index that applies only to this column and a `NOT NULL` constraint has been defined on the column ■ the column is an explicitly selected `ROWID` `IsUnique` is false if the column can contain duplicate values in the base table. The default is `false`. The value of this property is the same for each occurrence of the base table column in the select list.

Table 4–48 OracleDataReader SchemaTable (Cont.)

Name	Name Type	Description
IsKey	System.Boolean	Indicates whether the column is a key column. true if the column is one of a set of columns in the rowset that, taken together, uniquely identify the row. The set of columns with IsKey set to true must uniquely identify a row in the rowset. There is no requirement that this set of columns is a minimal set of columns. This set of columns can be generated from one of the following in descending order of priority: ■ A base table primary key. ■ Any of the unique constraints or unique indexes with the following condition: A NOT NULL constraint must be defined on the column or on all of the columns, in the case of a composite unique constraint or composite unique index. ■ Any of the composite unique constraints or composite unique indexes with the following condition: A NULL constraint must be defined on at least one, but not all, of the columns. An explicitly selected ROWID. False if the column is not required to uniquely identify the row. The value of this property is the same for each occurrence of the base table column in the select list.
IsRowID	System.Boolean	true if the column is a ROWID, otherwise false.
BaseColumnName	System.String	The name of the column in the database if an alias is used for the column.
BaseSchemaName	System.String	The name of the schema in the database that contains the column.
BaseTableName	System.String	The name of the table or view in the database that contains the column.
DataType	System.RuntimeType	Maps to the common language runtime type.
ProviderType	Oracle.DataAccess.Client.OracleDbType	The database column type (OracleDbType) of the column.
AllowDBNull	System.Boolean	true if null values are allowed, otherwise false.
IsAliased	System.Boolean	true if the column is an alias; otherwise false.

Table 4–48 OracleDataReader SchemaTable (Cont.)

Name	Name Type	Description
IsByteSemantic	System.Boolean	IsByteSemantic is: ■ true if the ColumnSize value uses bytes semantics ■ false if ColumnSize uses character semantics This value is always true when connected to a database version earlier than Oracle9*i*.
IsExpression	System.Boolean	true if the column is an expression; otherwise false.
IsHidden	System.Boolean	true if the column is hidden; otherwise false.
IsReadOnly	System.Boolean	true if the column is read-only; otherwise false.
IsLong	System.Boolean	true if the column is a LONG, LONG RAW, BLOB, CLOB, or BFILE; otherwise false.

Example

This example creates and uses the SchemaTable from the reader.

```
// C#
public static void ReadSchemaTable(string connStr)
 {
   .....

 //get the reader
 OracleDataReader reader = cmd.ExecuteReader();

 //get the schema table
 DataTable schemaTable = reader.GetSchemaTable();

 //retrieve the first column info.
 DataRow col0 = schemaTable.Rows[0];

 //print out the column info
 Console.WriteLine("Column name: " + col0["COLUMNNAME"]);
 Console.WriteLine("Precision: " + col0["NUMERICPRECISION"]);
 Console.WriteLine("Scale: " + col0["NUMERICSCALE"]);
 .....

 }
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetString

This method returns the `string` value of the specified column.

Declaration

```
// C#
public string GetString(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `string` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetTimeSpan

This method returns the `TimeSpan` value of the specified `INTERVAL DAY TO SECOND` column.

Declaration

```
// C#
public TimeSpan GetTimeSpan(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The `TimeSpan` value of the column.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetValue

This method returns the column value as a .NET type.

Declaration

```
// C#
public object GetValue(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

The value of the column as a .NET type.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

Remarks

When this method is invoked for a `NUMBER` column, the .NET type returned depends on the precision and scale of the column. For example, if a column is defined as `NUMBER(4,0)` then values in this column are retrieved as a `System.Int16`.

If the precision and scale is such that no .NET type can represent all the possible values that could exist in that column, the value is returned as a `System.Decimal`, if possible. If the value cannot be represented by a `System.Decimal`, an exception

is raised. For example, if a column is defined as `NUMBER (20,10)` then a value in this column is retrieved as a `System.Decimal`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetValues

This method gets all the column values as .NET types.

Declaration

```
// C#
public int GetValues(object[ ] values);
```

Parameters

- *values*

  An array of objects to hold the .NET types as the column values.

Return Value

The number of objects in the *values* array.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

Remarks

This method provides a way to retrieve all column values rather than retrieving each column value individually.

The number of column values retrieved is the minimum of the length of the values array and the number of columns in the result set.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

GetXmlReader

This method returns the contents of an `XMLType` column as an instance of an .NET `XmlTextReader` object.

Declaration

```
// C#
public XmlReader GetXmlReader(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

A .NET `XmlTextReader`.

Exceptions

`InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

IsDBNull

This method indicates whether the column value is `NULL`.

Declaration

```
// C#
public bool IsDBNull(int index);
```

Parameters

- *index*

  The zero-based column index.

Return Value

Returns `true` if the column is a `NULL` value; otherwise, returns `false`.

Implements

`IDataRecord`

Exceptions

`InvalidOperationException` - The reader is closed, `Read()` has not been called, or all rows have been read.

`IndexOutOfRangeException` - The column index is invalid.

Remarks

This method should be called to check for `NULL` values before calling the other accessor methods.

Example

The code example for the `OracleDataReader` class includes the `IsDBNull` method. See "Example" on page 4-113.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

NextResult

This method advances the data reader to the next result set.

Declaration

```
// C#
public bool NextResult();
```

Return Value

Returns `true` if another result set exists; otherwise, returns `false`.

Implements

`IDataReader`

Exceptions

`InvalidOperationException` - The connection is closed or the reader is closed.

Remarks

`NextResult` is used when reading results from stored procedure execution that return more than one result set.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

Read

This method reads the next row in the result set.

Declaration

```
// C#
public bool Read();
```

Return Value

Returns `true` if another row exists; otherwise, returns `false`.

Implements

`IDataReader`

Exceptions

`InvalidOperationException` - The connection is closed or the reader is closed.

Remarks

The initial position of the data reader is before the first row. Therefore, the `Read` method must be called to fetch the first row. The row that was just read is considered the *current row*. If the `OracleDataReader` has no more rows to read, it returns `false`.

Example

The code example for the `OracleDataReader` class includes the `Read` method. See "Example" on page 4-113.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleDataReader Class
- OracleDataReader Members

OracleError Class

The `OracleError` class represents an error reported by Oracle.

Class Inheritance

```
Object
  OracleError
```

Declaration

```
// C#
public sealed class OracleError
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The `OracleError` class represents a warning or an error reported by Oracle.

Example

```
// C#
...
try {
    cmd.ExecuteNonQuery()
}
catch ( OracleException e ){
   OracleError err1 = e.Errors[0];
   OracleError err2 = e.Errors[1];

    Console.WriteLine("Error 1 Message:", err1.Message);
    Console.WriteLine("Error 2 Source:", err2.Source);
}
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Members
- OracleError Static Methods
- OracleError Properties
- OracleError Methods

OracleError Members

`OracleError` members are listed in the following tables:

OracleError Static Methods

`OracleError` static methods are listed in Table 4–49.

Table 4–49 OracleError Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleError Properties

`OracleError` properties are listed in Table 4–50.

Table 4–50 OracleError Properties

Properties	Description
ArrayBindIndex	Specifies the row number of errors that occurred during the Array Bind execution
DataSource	Specifies the Oracle service name (TNS name) that identifies the Oracle database
Message	Specifies the `message` describing the `error`
Number	Specifies the Oracle `error` number
Procedure	Specifies the stored procedure that causes the `error`
Source	Specifies the name of the data provider that generates the `error`

OracleError Methods

`OracleError` methods are listed in Table 4–51.

Table 4–51 OracleError Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
ToString	Returns a string representation of the `OracleError`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class

OracleError Static Methods

`OracleError` static methods are listed in Table 4–52.

Table 4–52 OracleError Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

OracleError Properties

`OracleError` properties are listed in Table 4–53.

Table 4–53 OracleError Properties

Properties	Description
ArrayBindIndex	Specifies the row number of errors that occurred during the Array Bind execution

Table 4–53 OracleError Properties (Cont.)

Properties	Description
DataSource	Specifies the Oracle service name (TNS name) that identifies the Oracle database
Message	Specifies the `message` describing the `error`
Number	Specifies the Oracle `error` number
Procedure	Specifies the stored procedure that causes the `error`
Source	Specifies the name of the data provider that generates the `error`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

ArrayBindIndex

This property specifies the row number of errors that occurred during the Array Bind execution.

Declaration

```
// C#
public int ArrayBindIndex {get;}
```

Property Value

An `int` value that specifies the row number for errors that occurred during the Array Bind execution.

Remarks

Default = 0.

This property is used for Array Bind operations only.

`ArrayBindIndex` represents the zero-based row number at which the error occurred during an Array Bind operation. For example, if an array bind execution causes two errors on the 2nd and 4th operations, two `OracleError` objects appear in the `OracleErrorCollection` with the `ArrayBindIndex` property values 2 and 4 respectively.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members
- "Array Binding" on page 3-29

DataSource

This property specifies the Oracle service name (TNS name) that identifies the Oracle database.

Declaration

```
// C#
public string DataSource {get;}
```

Property Value

A `string`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

Message

This property specifies the `message` describing the `error`.

Declaration

```
// C#
public string Message {get;}
```

Property Value

A `string`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

Number

This property specifies the Oracle `error` number.

Declaration

```
// C#
public int Number {get;}
```

Property Value

An `int`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

Procedure

This property specifies the stored procedure that causes the `error`.

Declaration

```
// C#
public string Procedure {get;}
```

Property Value

The stored procedure name.

Remarks

Represents the stored procedure which creates this `OracleError` object.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

Source

This property specifies the name of the data provider that generates the `error`.

Declaration

```
// C#
public string Source {get;}
```

Property Value

A `string`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

OracleError Methods

`OracleError` methods are listed in Table 4–54.

Table 4–54 OracleError Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
ToString	Returns a string representation of the `OracleError`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

ToString

Overrides `Object`

This method returns a string representation of the `OracleError`.

Declaration

```
// C#
public override string ToString();
```

Return Value

Returns a string with the format Ora- `error number: Class.Method name error message stack trace information.`

Example

ORA-24333: zero iteration count

See Also:

- Oracle.DataAccess.Client Namespace
- OracleError Class
- OracleError Members

OracleErrorCollection Class

An `OracleErrorCollection` class represents a collection of all errors that are thrown by the Oracle Data Provider for .NET.

Class Inheritance

```
Object
  ArrayList
    OracleErrorCollection
```

Declaration

```
// C#
public sealed class OracleErrorCollection : ArrayList
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

A simple `ArrayList` that holds a list of `OracleErrors`.

Example

```
// C#
// The following example demonstrates how to access an individual OracleError
// from an OracleException
...
public void DisplayErrors(OracleException myException)
{
  for (int i=0; i < myException.Errors.Count; i++;)
  {
    Console.WriteLine("Index #" + i + "\n" +
      "Error: " + myException.Errors[i].ToString() + "\n");
  }
}
...
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleErrorCollection Members
- OracleErrorCollection Static Methods
- OracleErrorCollection Properties
- OracleErrorCollection Public Methods

OracleErrorCollection Members

`OracleErrorCollection` members are listed in the following tables:

OracleErrorCollection Static Methods

`OracleErrorCollection` static methods are listed in Table 4–55.

Table 4–55 OracleErrorCollection Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleErrorCollection Properties

`OracleErrorCollection` properties are listed in Table 4–56.

Table 4–56 OracleErrorCollection Properties

Name	Description
`Capacity`	Inherited from `ArrayList`
`Count`	Inherited from `ArrayList`
`IsReadOnly`	Inherited from `ArrayList`
`IsSynchronized`	Inherited from `ArrayList`
`Item`	Inherited from `ArrayList`

OracleErrorCollection Public Methods

`OracleErrorCollection` public methods are listed in Table 4–57.

Table 4–57 OracleErrorCollection Public Methods

Public Method	Description
`CopyTo`	Inherited from `ArrayList`
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleErrorCollection Class

OracleErrorCollection Static Methods

`OracleErrorCollection` static methods are listed in Table 4–58.

Table 4–58 OracleErrorCollection Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleErrorCollection Class
- OracleErrorCollection Members

OracleErrorCollection Properties

`OracleErrorCollection` properties are listed in Table 4–59.

Table 4–59 OracleErrorCollection Properties

Name	Description
`Capacity`	Inherited from `ArrayList`
`Count`	Inherited from `ArrayList`
`IsReadOnly`	Inherited from `ArrayList`
`IsSynchronized`	Inherited from `ArrayList`
`Item`	Inherited from `ArrayList`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleErrorCollection Class
- OracleErrorCollection Members

OracleErrorCollection Public Methods

`OracleErrorCollection` public methods are listed in Table 4–60.

Table 4–60 OracleErrorCollection Public Methods

Public Method	Description
`CopyTo`	Inherited from `ArrayList`
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleErrorCollection Class
- OracleErrorCollection Members

OracleException Class

The `OracleException` class represents an exception that is thrown when the Oracle Data Provider for .NET encounters an error. Each `OracleException` object contains at least one `OracleError` object in the `Error` property that describes the error or warning.

Class Inheritance

```
Object
  Exception
    SystemException
      OracleException
```

Declaration

```
// C#
public sealed class OracleException : SystemException
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
//  The following example generates an OracleException due to bad SQL syntax,
//  (that is the missing keyword "from") and then displays the exception message
//  and source property.
..
try
{
  ...
  // select * emp will cause ORA-00923
  OracleCommand cmd = new OracleCommand("select * emp", con);
}
catch ( OracleException e )
{
  Console.WriteLine("{0} throws {1}",e.Source, e.Message);
}
..
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Members
- OracleException Methods
- OracleException Static Methods
- OracleException Static Methods
- OracleException Properties
- OracleException Methods

OracleException Members

`OracleException` members are listed in the following tables:

OracleException Static Methods

`OracleException` static methods are listed in Table 4–61.

Table 4–61 OracleException Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleException Properties

`OracleException` properties are listed in Table 4–62.

Table 4–62 OracleException Properties

Properties	Description
DataSource	Specifies the TNS name that contains the information for connecting to an Oracle instance

Table 4–62 OracleException Properties (Cont.)

Properties	Description
Errors	Specifies a collection of one or more `OracleError` objects that contain information about exceptions generated by the Oracle database
`HelpLink`	Inherited from `Exception`
`InnerException`	Inherited from `Exception`
Message	Specifies the error messages that occur in the exception
Number	Specifies the Oracle error number
Procedure	Specifies the stored procedure that cause the exception
Source	Specifies the name of the data provider that generates the error
`StackTrace`	Inherited from `Exception`
`TargetSite`	Inherited from `Exception`

OracleException Methods

`OracleException` methods are listed in Table 4–63.

Table 4–63 OracleException Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetBaseException`	Inherited from `Exception`
`GetHashCode`	Inherited from `Object`
GetObjectData	Sets the serializable `info` object with information about the exception
`GetType`	Inherited from `Object`
ToString	Returns the fully qualified name of this exception

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class

OracleException Static Methods

`OracleException` static methods are listed in Table 4–64.

Table 4–64 OracleException Static Methods

Method	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

OracleException Properties

`OracleException` properties are listed in Table 4–65.

Table 4–65 OracleException Properties

Properties	Description
DataSource	Specifies the TNS name that contains the information for connecting to an Oracle instance
Errors	Specifies a collection of one or more `OracleError` objects that contain information about exceptions generated by the Oracle database
`HelpLink`	Inherited from `Exception`
`InnerException`	Inherited from `Exception`
Message	Specifies the error messages that occur in the exception
Number	Specifies the Oracle error number
Procedure	Specifies the stored procedure that cause the exception
Source	Specifies the name of the data provider that generates the error
`StackTrace`	Inherited from `Exception`
`TargetSite`	Inherited from `Exception`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

DataSource

This property specifies the TNS name that contains the information for connecting to an Oracle instance.

Declaration

```
// C#
public string DataSource {get;}
```

Property Value

The TNS name containing the connect information.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

Errors

This property specifies a collection of one or more `OracleError` objects that contain information about exceptions generated by the Oracle database.

Declaration

```
// C#
public OracleErrorCollection Errors {get;}
```

Property Value

An `OracleErrorCollection`.

Remarks

The `Errors` property contains at least one instance of `OracleError` objects.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

Message

Overrides `Exception`

This property specifies the error messages that occur in the exception.

Declaration

```
// C#
public override string Message {get;}
```

Property Value

A `string`.

Remarks

`Message` is a concatenation of all errors in the `Errors` collection. Each error message is concatenated and is followed by a carriage return, except the last one.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

Number

This property specifies the Oracle error number.

Declaration

```
// C#
public int Number {get;}
```

Property Value

The error number.

Remarks

This error number can be the topmost level of error generated by Oracle and can be a provider-specific error number.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

Procedure

This property specifies the stored procedure that caused the exception.

Declaration

```
// C#
public string Procedure {get;}
```

Property Value

The stored procedure name.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

Source

Overrides `Exception`

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#
public override string Source {get;}
```

Property Value

The name of the data provider.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

OracleException Methods

`OracleException` methods are listed in Table 4–66.

Table 4–66 OracleException Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetBaseException`	Inherited from `Exception`
`GetHashCode`	Inherited from `Object`
GetObjectData	Sets the serializable `info` object with information about the exception
`GetType`	Inherited from `Object`
ToString	Returns the fully qualified name of this exception

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

GetObjectData

Overrides `Exception`

This method sets the serializable `info` object with information about the exception.

Declaration

```
// C#
public override void GetObjectData(SerializationInfo info, StreamingContext
context);
```

Parameters

- *info*

  A `SerializationInfo` object.

- *context*

  A `StreamingContext` object.

Remarks

The information includes `DataSource`, `Message`, `Number`, `Procedure`, `Source`, and `StackTrace`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

ToString

Overrides `Exception`

This method returns the fully qualified name of this exception, the `error` message in the `Message` property, the `InnerException.ToString()` message, and the stack trace.

Declaration

```
// C#
public override string ToString();
```

Return Value

The string representation of the exception.

Example

```
// C#
...
try
{
  ...
  // select * from emp will cause ORA-00923
  OracleCommand cmd = new OracleCommand("select * from emp", con);
}
```

```
catch ( OracleException e )
{
  Console.WriteLine("{0}",e.ToString());
}
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleException Class
- OracleException Members

OracleFailoverEventArgs Class

The `OracleFailoverEventArgs` class provides event data for the `OracleConnection.Failover` event. When database failover occurs, the `OracleConnection.Failover` event is triggered along with the `OracleFailoverEventArgs` object that stores the event data.

Class Inheritance

```
Object
  EventArgs
    OracleFailoverEventArgs
```

Declaration

```
// C#
public sealed class OracleFailoverEventArgs
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
// Receiving Failover notifications
switch (eventArgs.FailoverEvent)
  {
    case FailoverEvent.Begin:
      {
       Console.WriteLine(" \nFailover Begin - Failing Over ..." +
          "Please stand by \n");
       Console.WriteLine(" \nFailover type was found to be " +
          eventArgs.FailoverType);
       break;
      }

    case FailoverEvent.End:
      {
       Console.WriteLine(" \nFailover ended ...resuming services\n");
       break;
      }
```

```
    case FailoverEvent.Error:
     {
      Console.WriteLine(" Failover error gotten. Sleeping...\n");
      Thread.Sleep(3000);
      return FailoverReturnCode.Retry;
     }

    default:
     {
      Console.WriteLine("\nBad Failover Event: " + eventArgs.FailoverEvent);
      break;
     }
  }
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleFailoverEventArgs Members
- OracleFailoverEventArgs Static Methods
- OracleFailoverEventArgs Properties
- OracleFailoverEventArgs Public Methods
- "OracleConnection Class" on page 4-54

OracleFailoverEventArgs Members

`OracleFailoverEventArgs` members are listed in the following tables:

OracleFailoverEventArgs Static Methods

The `OracleFailoverEventArgs` static methods are listed in Table 4–67.

Table 4–67 OracleFailoverEventArgs Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleFailoverEventArgs Properties

The `OracleFailoverEventArgs` properties are listed in Table 4–68.

Table 4–68 OracleFailoverEventArgs Properties

Name	Description
FailoverType	Specifies the type of failover the client has requested
FailoverEvent	Indicates the state of the failover

OracleFailoverEventArgs Public Methods

The `OracleFailoverEventArgs` public methods are listed in Table 4–69.

Table 4–69 OracleFailoverEventArgs Public Methods

Name	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleFailoverEventArgs Class
- "FailoverType Enumeration" on page 4-359

OracleFailoverEventArgs Static Methods

The `OracleFailoverEventArgs` static methods are listed in Table 4–70.

Table 4–70 OracleFailoverEventArgs Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members

OracleFailoverEventArgs Properties

The `OracleFailoverEventArgs` properties are listed in Table 4–71.

Table 4–71 OracleFailoverEventArgs Properties

Name	Description
FailoverType	Specifies the type of failover the client has requested
FailoverEvent	Indicates the state of the failover

See Also:

- Oracle.DataAccess.Client Namespace
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members

FailoverType

This property indicates the state of the failover.

Declaration

```
// C#
public FailoverType FailoverType {get;}
```

Property Value

A `FailoverType` enumeration value.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members
- "FailoverType Enumeration" on page 4-359

FailoverEvent

This property indicates the state of the failover.

Declaration

```
// C#
public FailoverEvent FailoverEvent {get;}
```

Property Value

A `FailoverEvent` enumerated value.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members
- "FailoverEvent Enumeration" on page 4-357

OracleFailoverEventArgs Public Methods

The `OracleFailoverEventArgs` public methods are listed in Table 4–72.

Table 4–72 OracleFailoverEventArgs Public Methods

Name	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members

OracleFailoverEventHandler Delegate

The `OracleFailoverEventHandler` represents the signature of the method that handles the `OracleConnection.Failover` event.

Declaration

```
// C#
public delegate FailoverReturnCode OracleFailoverEventHandler(object sender,
OracleFailoverEventArgs eventArgs);
```

Parameter

- *sender*

  The source of the event.

- *eventArgs*

  The `OracleFailoverEventArgs` object that contains the event data.

Return Type

An `int`.

Remarks

To receive failover notifications, a callback function can be registered as follows:

```
ConObj.Failover += new OracleFailoverEventHandler(OnFailover);
```

The definition of the callback function `OnFailover` can be as follows:

```
public FailoverReturnCode OnFailover(object sender, OracleFailoverEventArgs
eventArgs)
```

Example

```
void Main(string[] args)
{
...
// register callback function OnFailOver
ConObj.Failover += new OracleFailoverEventHandler(OnFailOver);
...
}
```

```
//Failover Callback Function
public FailoverReturnCode OnFailOver(object sender, OracleFailoverEventArgs
eventArgs)
{

      switch (eventArgs.FailoverEvent)
      {
        case FailoverEvent.Begin:
        {
          Console.WriteLine(" \nFailover Begin - Failing Over ... Please stand
                            by \n");
          Console.WriteLine(" Failover type was found to be " +
                            eventArgs.FailoverType);
          break;
        }

        case FailoverEvent.Abort:
        {
          Console.WriteLine(" Failover aborted. Failover will not take
                             place.\n");
          break;
        }

        case FailoverEvent.End:
        {
          Console.WriteLine(" Failover ended ...resuming services\n");
          break;
        }

        case FailoverEvent.Reauth:
        {
          Console.WriteLine(" Failed over user. Resuming services\n");
          break;
        }

        case FailoverEvent.Error:
        {
          Console.WriteLine(" Failover error gotten. Sleeping...\n");
          Thread.Sleep(3000);
          return FailoverReturnCode.Retry;
        }

        default:
        {
          Console.WriteLine("Bad Failover Event: %d.\n",
```

```
                                eventArgs.FailoverEvent);
            break;
          }
        }

        return FailoverReturnCode.Success;

} /* OnFailover */
```

See Also:

- Oracle.DataAccess.Client Namespace
- "OracleConnection Class" on page 4-54

OracleGlobalization Class

The `OracleGlobalization` class is used to obtain and set the Oracle globalization settings of the session, thread, and local computer (read-only).

Class Inheritance

```
Object
  OracleGlobalization
```

Declaration

```
public sealed class OracleGlobalization : ICloneable, IDisposable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

An exception is thrown for invalid property values. All newly set property values are validated, except the `TimeZone` property.

Changing the `OracleGlobalization` object properties does not change the globalization settings of the session or the thread. Either the `SetSessionInfo` method of the `OracleConnection` object or the `SetThreadInfo` method of the `OracleGlobalization` object must be called to alter the session's and thread's globalization settings, respectively.

Example

```
// C#
//  Sets thread globalization info.
...
string ConStr = "User Id=myschema;Password=mypassword;" +
   "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Retrieves thread globalization info
OracleGlobalization ogi = OracleGlobalization.GetThreadInfo();

//Print the language name in thread globalization info
Console.WriteLine("Thread language: " + ogi.Language);
```

```
//Set thread's language
ogi.Language = "FRENCH";
OracleGlobalization.SetThreadInfo(ogi);

OracleGlobalization ogi2;
OracleGlobalization.GetThreadInfo(ogi2);

//Print the language name in thread globalization info
Console.WriteLine("Thread language: " + ogi2.Language);
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Members
- OracleGlobalization Static Methods
- OracleGlobalization Properties
- OracleGlobalization Public Methods
- *Oracle Database SQL Reference*
- *Oracle Database Globalization Support Guide*
- "Globalization Support" on page 3-39

OracleGlobalization Members

`OracleGlobalization` members are listed in the following tables:

OracleGlobalization Static Methods

The `OracleGlobalization` static methods are listed in Table 4–73.

Table 4–73 OracleGlobalization Static Methods

Name	Description
GetClientInfo	Returns an `OracleGlobalization` object that represents the Oracle globalization settings of the local computer (Overloaded)
GetThreadInfo	Returns or refreshes an `OracleGlobalization` instance that represents Oracle globalization settings of the current thread (Overloaded)
SetThreadInfo	Sets Oracle globalization parameters to the current thread

OracleGlobalization Properties

The `OracleGlobalization` properties are listed in Table 4–74.

Table 4–74 OracleGlobalization Properties

Name	Description
Calendar	Specifies the calendar system
ClientCharacterSet	Specifies a client character set
Comparison	Specifies a method of comparison for `WHERE` clauses and comparison in PL/SQL blocks
Currency	Specifies the string to use as a local currency symbol for the L number format element
DateFormat	Specifies the date format for Oracle `Date` type as a string
DateLanguage	Specifies the language used to spell day and month names and date abbreviations
DualCurrency	Specifies the dual currency symbol, such as *Euro*, for the U number format element
ISOCurrency	Specifies the string to use as an international currency symbol for the C number format element
Language	Specifies the default language of the database
LengthSemantics	Enables creation of `CHAR` and `VARCHAR2` columns using either byte or character (default) length semantics
NCharConversionException	Determines whether data loss during an implicit or explicit character type conversion reports an error

Table 4–74 OracleGlobalization Properties (Cont.)

Name	Description
NumericCharacters	Specifies the characters used for the decimal character and the group separator character for numeric values in strings
Sort	Specifies the collating sequence for `ORDER` by clause
Territory	Specifies the name of the territory
TimeStampFormat	Specifies the string format for `TimeStamp` types
TimeStampTZFormat	Specifies the string format for `TimeStampTZ` types
TimeZone	Specifies the time zone region name

OracleGlobalization Public Methods

`OracleGlobalization` public methods are listed in Table 4–75.

Table 4–75 OracleGlobalization Public Methods

Public Method	Description
Clone	Creates a copy of an `OracleGlobalization` object
`Dispose`	Inherited from `Component`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class

OracleGlobalization Static Methods

The `OracleGlobalization` static methods are listed in Table 4–76.

Table 4–76 OracleGlobalization Static Methods

Name	Description
GetClientInfo	Returns an `OracleGlobalization` object that represents the Oracle globalization settings of the local computer (Overloaded)
GetThreadInfo	Returns or refreshes an `OracleGlobalization` instance that represents Oracle globalization settings of the current thread (Overloaded)

Table 4–76 OracleGlobalization Static Methods (Cont.)

Name	Description
SetThreadInfo	Sets Oracle globalization parameters to the current thread

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

GetClientInfo

`GetClientInfo` returns an `OracleGlobalization` object instance that represents the Oracle globalization settings of the local computer.

Overload List:

- GetClientInfo()

  This method returns an `OracleGlobalization` instance that represents the globalization settings of the local computer.

- GetClientInfo(OracleGlobalization)

  This method refreshes the provided `OracleGlobalization` object with the globalization settings of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

GetClientInfo()

This method returns an `OracleGlobalization` instance that represents the globalization settings of the local computer.

Declaration

```
// C#
public static OracleGlobalization GetClientInfo();
```

Return Value

An `OracleGlobalization` instance.

Example

```
// C#
//  Retrieves the client globalization info.
...
string ConStr = "User Id=myschema;Password=mypassword;" +
     "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Retrieves the client globalization info
OracleGlobalization ogi = OracleGlobalization.GetClientInfo();

//Retrieves the client globalization info using overloaded method
OracleGlobalization ogi2;
OracleGlobalization.GetClientInfo(ogi2);

//Print the language name in client globalization info
Console.WriteLine("Client machine language: " + ogi.Language);
Console.WriteLine("Client machine language: " + ogi2.Language);
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

GetClientInfo(OracleGlobalization)

This method refreshes the provided `OracleGlobalization` object with the globalization settings of the local computer.

Declaration

```
// C#
public static void GetClientInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

The `OracleGlobalization` object being updated.

Example

```
// C#
//  Retrieves the client globalization info.
...
string ConStr = "User Id=myschema;Password=mypassword;" +
     "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Retrieves the client globalization info
OracleGlobalization ogi = OracleGlobalization.GetClientInfo();

//Retrieves the client globalization info using overloaded method
OracleGlobalization ogi2;
OracleGlobalization.GetClientInfo(ogi2);

//Print the language name in client globalization info
Console.WriteLine("Client machine language: " + ogi.Language);
Console.WriteLine("Client machine language: " + ogi2.Language);
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

GetThreadInfo

`GetThreadInfo` returns or refreshes an `OracleGlobalization` instance.

Overload List:

- GetThreadInfo()

  This method returns an `OracleGlobalization` object instance of the current thread.

- GetThreadInfo(OracleGlobalization)

  This method refreshes the `OracleGlobalization` object instance with the globalization settings of the current thread.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

GetThreadInfo()

This method returns an `OracleGlobalization` instance of the current thread.

Declaration

```
// C#
public static OracleGlobalization GetThreadInfo();
```

Return Value

An `OracleGlobalization` instance.

Remarks

Initially, `GetThreadInfo()` returns an `OracleGlobalization` object that has the same property values as that returned by `GetClientInfo()`, unless the application changes it by invoking `SetThreadInfo()`.

Example

```
// C#
Retrieves the thread globalization info.
...
string ConStr = "User Id=myschema;Password=mypassword;" +
       "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Retrieves the thread globalization info
OracleGlobalization ogi = OracleGlobalization.GetThreadInfo();

//Retrieves the thread globalization info using overloaded method
OracleGlobalization ogi2;
OracleGlobalization.GetThreadInfo(ogi2);

//Print the language name in thread globalization info
Console.WriteLine("Thread language: " + ogi.Language);
Console.WriteLine("Thread language: " + ogi2.Language);
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

GetThreadInfo(OracleGlobalization)

This method refreshes the `OracleGlobalization` object with the globalization settings of the current thread.

Declaration

```
// C#
public static void GetThreadInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

  The `OracleGlobalization` object being updated.

Remarks

Initially `GetThreadInfo()` returns an `OracleGlobalization` object that has the same property values as that returned by `GetClientInfo()`, unless the application changes it by invoking `SetThreadInfo()`.

Example

```
// C#
Retrieves the thread globalization info.
...
string ConStr = "User Id=myschema;Password=mypassword;" +
       "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Retrieves the thread globalization info
OracleGlobalization ogi = OracleGlobalization.GetThreadInfo();

//Retrieves the thread globalization info using overloaded method
OracleGlobalization ogi2;
OracleGlobalization.GetThreadInfo(ogi2);
```

```
//Print the language name in thread globalization info
Console.WriteLine("Thread language: " + ogi.Language);
Console.WriteLine("Thread language: " + ogi2.Language);
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

SetThreadInfo

This method sets Oracle globalization parameters to the current thread.

Declaration

```
// C#
public static void SetThreadInfo(OracleGlobalization oraGlob);
```

Parameters

- *oraGlob*

  An `OracleGlobalization` object.

Remarks

Any .NET string conversions to and from ODP.NET Types, as well as ODP.NET Type constructors, use the globalization property values where applicable. For example, when constructing an `OracleDate` structure from a .NET string, that string is expected to be in the format specified by the `OracleGlobalization.DateFormat` property of the thread.

Example

```
// C#
//  Sets thread globalization info.
...
string ConStr = "User Id=myschema;Password=mypassword;" +
   "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Retrieves thread globalization info
```

```
OracleGlobalization ogi = OracleGlobalization.GetThreadInfo();

//Print the language name in thread globalization info
Console.WriteLine("Thread language: " + ogi.Language);

//Set thread's language
ogi.Language = "FRENCH";
OracleGlobalization.SetThreadInfo(ogi);

OracleGlobalization ogi2;
OracleGlobalization.GetThreadInfo(ogi2);

//Print the language name in thread globalization info
Console.WriteLine("Thread language: " + ogi2.Language);
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

OracleGlobalization Properties

The `OracleGlobalization` properties are listed in Table 4–77.

Table 4–77 OracleGlobalization Properties

Name	Description
Calendar	Specifies the calendar system
ClientCharacterSet	Specifies a client character set
Comparison	Specifies a method of comparison for `WHERE` clauses and comparison in PL/SQL blocks
Currency	Specifies the string to use as a local currency symbol for the L number format element
DateFormat	Specifies the date format for Oracle `Date` type as a string
DateLanguage	Specifies the language used to spell day and month names and date abbreviations
DualCurrency	Specifies the dual currency symbol, such as *Euro*, for the U number format element

Table 4–77 OracleGlobalization Properties (Cont.)

Name	Description
ISOCurrency	Specifies the string to use as an international currency symbol for the C number format element
Language	Specifies the default language of the database
LengthSemantics	Enables creation of `CHAR` and `VARCHAR2` columns using either byte or character (default) length semantics
NCharConversionException	Determines whether data loss during an implicit or explicit character type conversion reports an error
NumericCharacters	Specifies the characters used for the decimal character and the group separator character for numeric values in strings
Sort	Specifies the collating sequence for `ORDER` by clause
Territory	Specifies the name of the territory
TimeStampFormat	Specifies the string format for `TimeStamp` types
TimeStampTZFormat	Specifies the string format for `TimeStampTZ` types
TimeZone	Specifies the time zone region name

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

Calendar

This property specifies the calendar system.

Declaration

```
// C#
public string Calendar {get; set;}
```

Property Value

A string representing the `Calendar`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_CALENDAR` setting of the local computer. This value is the same regardless of whether the `OracleGlobalization` object represents the settings of the client, thread, or session.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

ClientCharacterSet

This property specifies a client character set.

Declaration

```
// C#
public string ClientCharacterSet {get;}
```

Property Value

A string that the provides the name of the character set of the local computer.

Remarks

The default value is the character set of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

Comparison

This property represents a method of comparison for `WHERE` clauses and comparison in PL/SQL blocks.

Declaration

```
// C#
public string Comparison {get; set;}
```

Property Value

A string that provides the name of the method of comparison.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_COMP` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

Currency

This property specifies the string to use as a local currency symbol for the L number format element.

Declaration

```
// C#
public string Currency {get; set;}
```

Property Value

The string to use as a local currency symbol for the L number format element.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_CURRENCY` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members
- *Oracle Database SQL Reference*for further information on the L number format element

DateFormat

This property specifies the date format for Oracle `Date` type as a string.

Declaration

```
// C#
public string DateFormat {get; set;}
```

Property Value

The date format for Oracle `Date` type as a string

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_DATE_FORMAT` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

DateLanguage

This property specifies the language used to spell names of days and months, and date abbreviations (for example: a.m., p.m., AD, BC).

Declaration

```
// C#
public string DateLanguage {get; set;}
```

Property Value

A string specifying the language.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_DATE_LANGUAGE` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

DualCurrency

This property specifies the dual currency symbol, such as *Euro*, for the U number format element.

Declaration

```
// C#
public string DualCurrency {get; set;}
```

Property Value

A string that provides the dual currency symbol.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_DUAL_CURRENCY` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members
- *Oracle Database SQL Reference* for further information on the U number format element

ISOCurrency

This property specifies the string to use as an international currency symbol for the C number format element.

Declaration

```
// C#
public string ISOCurrency {get; set;}
```

Property Value

The string used as an international currency symbol.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_ISO_CURRENCY` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members
- *Oracle Database SQL Reference* for further information on the C number format element

Language

This property specifies the default language of the database.

Declaration

```
// C#
public string Language {get; set;}
```

Property Value

The default language of the database.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_LANGUAGE` setting of the local computer.

`Language` is used for messages, day and month names, and sorting algorithms. It also determines `NLS_DATE_LANGUAGE` and `NLS_SORT` parameter values.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

LengthSemantics

This property indicates whether `CHAR` and `VARCHAR2` columns use byte or character (default) length semantics.

Declaration

```
// C#
public string LengthSemantics {get; set;}
```

Property Value

A string that indicates either byte or character length semantics.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_LENGTH_SEMANTICS` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

NCharConversionException

This property determines whether data loss during an implicit or explicit character type conversion reports an error.

Declaration

```
// C#
public bool NCharConversionException {get; set;}
```

Property Value

A string that indicates whether or not a character type conversion causes an error message.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_NCHAR_CONV_EXCP` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

NumericCharacters

This property specifies the characters used for the decimal character and the group separator character for numeric values in strings.

Declaration

```
// C#
public string NumericCharacters {get; set;}
```

Property Value

A string that represents the characters used.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_NUMERIC_CHARACTERS` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

Sort

This property specifies the collating sequence for `ORDER` by clause.

Declaration

```
// C#
public string Sort {get; set;}
```

Property Value

A string that indicates the collating sequence.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_SORT` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

Territory

This property specifies the name of the territory.

Declaration

```
// C#
public string Territory {get; set;}
```

Property Value

A string that provides the name of the territory.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_TERRITORY` setting of the local computer.

Changing this property changes other globalization properties.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members
- *Oracle Database Globalization Support Guide*.

TimeStampFormat

This property specifies the string format for `TimeStamp` types.

Declaration

```
// C#
public string TimeStampFormat {get; set;}
```

Property Value

The string format for `TimeStamp` types.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_TIMESTAMP_FORMAT` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

TimeStampTZFormat

This property specifies the string format for `TimeStampTZ` types.

Declaration

```
// C#
public string TimeStampTZFormat {get; set;}
```

Property Value

The string format for `TimeStampTZ` types.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the `NLS_TIMESTAMP_TZ_FORMAT` setting of the local computer.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

TimeZone

This property specifies the time zone region name or hour offset.

Declaration

```
// C#
public string TimeZone {get; set;}
```

Property Value

The string represents the time zone region name or the time zone offset.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The default value is the time zone region name of the local computer

`TimeZone` is only used when the thread constructs one of the `TimeStamp` structures. `TimeZone` has no effect on the session.

`TimeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

Note: PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by `OracleGlobalization`.

This property returns an empty string if the `OracleGlobalization` object is obtained using `GetSessionInfo()` or `GetSessionInfo(OracleGlobalization)`. Initially, by default, the time zone of the session is identical to the time zone of the thread. Therefore, given that the session time zone is not changed by invoking `ALTER SESSION` calls, the session time zone can be fetched from the client's globalization settings.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

OracleGlobalization Public Methods

`OracleGlobalization` public methods are listed in Table 4–78.

Table 4–78 OracleGlobalization Public Methods

Public Method	Description
Clone	Creates a copy of an `OracleGlobalization` object
`Dispose`	Inherited from `Component`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

Clone

This method creates a copy of an `OracleGlobalization` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleGlobalization` object.

Implements

`ICloneable`

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
//Need a proper casting for the return value when cloned
OracleGlobalization ogi_cloned = (OracleGlobalization) ogi.Clone();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleGlobalization Class
- OracleGlobalization Members

OracleInfoMessageEventArgs Class

The `OracleInfoMessageEventArgs` class provides event data for the `OracleConnection.InfoMessage` event. When any warning occurs in the database, the `OracleConnection.InfoMessage` event is triggered along with the `OracleInfoMessageEventArgs` object that stores the event data.

Class Inheritance

```
Object

  EventArgs

    OracleInfoMessageEventArgs
```

Declaration

```
// C#
public sealed class OracleInfoMessageEventArgs
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
public void WarningHandler(object src, OracleInfoMessageEventArgs args)
    {
      LogOutput("Source object is: " + src.GetType().Name);
      LogOutput("InfoMessageArgs.Message is " + args.Message);
      LogOutput("InfoMessageArgs.Errors is " + args.Errors);
      LogOutput("InfoMessageArgs.Source is " + args.Source);
    }

    public bool MyFunc()
    {
      ...
      con.Open();
      OracleCommand cmd = Con.CreateCommand();

      //Register to the InfoMessageHandler
      cmd.Connection.InfoMessage +=
        new OracleInfoMessageEventHandler(WarningHandler);
```

```
    cmd.CommandText = CmdStr;
    cmd.CommandType = CommandType.Text;
    //If CmdStr causes warning(s), it will be handled.
    cmd.ExecuteNonQuery();
    ...
  }
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleInfoMessageEventArgs Members
- OracleInfoMessageEventArgs Static Methods
- OracleInfoMessageEventArgs Properties
- OracleInfoMessageEventArgs Public Methods
- "OracleConnection Class" on page 4-54

OracleInfoMessageEventArgs Members

`OracleInfoMessageEventArgs` members are listed in the following tables:

OracleInfoMessageEventArgs Static Methods

The `OracleInfoMessageEventArgs` static methods are listed in Table 4–79.

Table 4–79 OracleInfoMessageEventArgs Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleInfoMessageEventArgs Properties

The `OracleInfoMessageEventArgs` properties are listed in Table 4–80.

Table 4–80 OracleInfoMessageEventArgs Properties

Name	Description
Errors	Specifies the collection of errors generated by the data source
Message	Specifies the error text generated by the data source
Source	Specifies the name of the object that generated the error

OracleInfoMessageEventArgs Public Methods

The `OracleInfoMessageEventArgs` methods are listed in Table 4–81.

Table 4–81 OracleInfoMessageEventArgs Public Methods

Name	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleInfoMessageEventArgs Class

OracleInfoMessageEventArgs Static Methods

The `OracleInfoMessageEventArgs` static methods are listed in Table 4–82.

Table 4–82 OracleInfoMessageEventArgs Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members

OracleInfoMessageEventArgs Properties

The `OracleInfoMessageEventArgs` properties are listed in Table 4–83.

Table 4–83 OracleInfoMessageEventArgs Properties

Name	Description
Errors	Specifies the collection of errors generated by the data source
Message	Specifies the error text generated by the data source
Source	Specifies the name of the object that generated the error

See Also:

- Oracle.DataAccess.Client Namespace
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members

Errors

This property specifies the collection of errors generated by the data source.

Declaration

```
// C#
public OracleErrorCollection Errors {get;}
```

Property Value

The collection of errors.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members

Message

This property specifies the error text generated by the data source.

Declaration

```
// C#
```

```
public string Message {get;}
```

Property Value

The error text.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members

Source

This property specifies the name of the object that generated the error.

Declaration

```
// C#
public string Source {get;}
```

Property Value

The object that generated the error.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members

OracleInfoMessageEventArgs Public Methods

The `OracleInfoMessageEventArgs` methods are listed in Table 4–84.

Table 4–84 OracleInfoMessageEventArgs Public Methods

Name	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members

OracleInfoMessageEventHandler Delegate

The `OracleInfoMessageEventHandler` represents the signature of the method that handles the `OracleConnection.InfoMessage` event.

Declaration

```
// C#
public delegate void OracleInfoMessageEventHandler(object sender,
   OracleInfoMessageEventArgs eventArgs);
```

Parameter

- *`sender`*

  The source of the event.

- *`eventArgs`*

  The `OracleInfoMessageEventArgs` object that contains the event data.

See Also:

- Oracle.DataAccess.Client Namespace
- "OracleConnection Class" on page 4-54

OracleParameter Class

An `OracleParameter` object represents a parameter for an `OracleCommand` or a `DataSet` column.

Class Inheritance

```
Object
  MarshalByRefObject
    OracleParameter
```

Declaration

```
// C#
public sealed class OracleParameter : MarshalByRefObject, IDBDataParameter,
   IDataParameter, IDisposable, ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Exceptions

`ArgumentException` - The type binding is invalid.

Example

```
// C#
...
OracleParameter [] prm = new OracleParameter[3];

// Create OracleParameter objects through OracleParameterCollection
prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal, 1234,
    ParameterDirection.Input);
prm[1] = cmd.Parameters.Add("paramEname", OracleDbType.Varchar2,
    "Client", ParameterDirection.Input);
prm[2] = cmd.Parameters.Add("paramDeptNo", OracleDbType.Decimal,
    10, ParameterDirection.Input);

cmd.CommandText = "insert into emp(empno, ename, deptno) values(:1, :2, :3)";
cmd.CommandType = CommandType.CommandText;
cmd.ExecuteNonQuery();
...
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Members
- OracleParameter Constructors
- OracleParameter Static Methods
- OracleParameter Properties
- OracleParameter Public Methods

OracleParameter Members

`OracleParameter` members are listed in the following tables:

OracleParameter Constructors

`OracleParameter` constructors are listed in Table 4–85.

Table 4–85 OracleParameter Constructors

Constructor	Description
OracleParameter Constructors	Instantiates a new instance of `OracleParameter` class (Overloaded)

OracleParameter Static Methods

`OracleParameter` static methods are listed in Table 4–86.

Table 4–86 OracleParameter Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleParameter Properties

`OracleParameter` properties are listed in Table 4–87.

Table 4–87 OracleParameter Properties

Name	Description
ArrayBindSize	Specifies the input or output size of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
ArrayBindStatus	Specifies the input or output status of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
CollectionType	Specifies whether the `OracleParameter` represents a collection, and if so, specifies the collection type
DbType	Specifies the datatype of the parameter using the `Data.DbType` enumeration type
Direction	Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter
`IsNullable`	*This method is a no-op*
Offset	Specifies the offset to the `Value` property or offset to the elements in the `Value` property
OracleDbType	Specifies the Oracle datatype
ParameterName	Specifies the name of the parameter
Precision	Specifies the maximum number of digits used to represent the `Value` property
Scale	Specifies the number of decimal places to which `Value` property is resolved
Size	Specifies the maximum size, in bytes or characters, of the data transmitted to or from the server. For PL/SQL Associative Array Bind, `Size` specifies the maximum number of elements in PL/SQL Associative Array.
SourceColumn	Specifies the name of the `DataTable` Column of the `DataSet`
SourceVersion	Specifies the `DataRowVersion` value to use when loading the `Value` property of the parameter
Status	Indicates the status of the execution related to the data in the `Value` property
Value	Specifies the value of the `Parameter`

OracleParameter Public Methods

`OracleParameter` public methods are listed in Table 4–88.

Table 4–88 OracleParameter Public Methods

Public Method	Description
Clone	Creates a shallow copy of an `OracleParameter` object
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Releases allocated resources
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
`ToString`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class

OracleParameter Constructors

`OracleParameter` constructors instantiate new instances of the `OracleParameter` class.

Overload List:

- OracleParameter()

  This constructor instantiates a new instance of `OracleParameter` class.

- OracleParameter (string, OracleDbType)

  This constructor instantiates a new instance of `OracleParameter` class using the supplied parameter name and Oracle datatype.

- OracleParameter(string, object)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name and parameter value.

- OracleParameter(string, OracleDbType, ParameterDirection)

  This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, and parameter direction.

- OracleParameter(string, OracleDbType, object, ParameterDirection)

  This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, value, and direction.

- OracleParameter(string, OracleDbType, int)

  This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, and size.

- OracleParameter(string, OracleDbType, int, string)

  This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, size, and source column.

- OracleParameter(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)

  This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, size, direction, null indicator, precision, scale, source column, source version and parameter value.

- OracleParameter(string, OracleDbType, int, object, ParameterDirection)

  This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, size, value, and direction.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

OracleParameter()

This constructor instantiates a new instance of `OracleParameter` class.

Declaration

```
// C#
```

```
public OracleParameter();
```

Remarks

Default Values:

- `DbType` - `String`
- `ParameterDirection` - `Input`
- `isNullable` - `true`
- `offset` - `0`
- `OracleDbType` - `Varchar2`
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363
- "OracleParameterCollection Class" on page 4-281

OracleParameter (string, OracleDbType)

This constructor instantiates a new instance of `OracleParameter` class using the supplied parameter name and Oracle datatype.

Declaration

```
// C#
```

```
public OracleParameter(string parameterName, OracleDbType oraType);
```

Parameters

- *parameterName*

  Specifies the parameter name.

- *oraType*

  Specifies the datatype of the `OracleParameter`.

Remarks

Changing the `DbType` implicitly changes the `OracleDbType`.

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType` - `String`
- `ParameterDirection` - `Input`
- `isNullable` - `true`
- `offset` - `0`
- `OracleDbType` - `Varchar2`
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363
- "OracleParameterCollection Class" on page 4-281

OracleParameter(string, object)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name and parameter value.

Declaration

```
// C#
public OracleParameter(string parameterName, object obj);
```

Parameters

- *parameterName*

  Specifies parameter name.

- *obj*

  Specifies value of the `OracleParameter`.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType` - `String`
- `ParameterDirection` - `Input`
- `isNullable` - `true`
- `offset` - `0`
- `OracleDbType` - `Varchar2`
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string

- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363
- "OracleParameterCollection Class" on page 4-281

OracleParameter(string, OracleDbType, ParameterDirection)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, and parameter direction.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType type,
ParameterDirection direction);
```

Parameters

- *`parameterName`*

  Specifies the parameter name.

- *`type`*

  Specifies the datatype of the `OracleParameter`.

- *`direction`*

  Specifies the direction of the `OracleParameter`.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType` - `String`
- `ParameterDirection` - `Input`
- `isNullable` - `true`
- `offset` - `0`
- `OracleDbType` - `Varchar2`
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363
- "OracleParameterCollection Class" on page 4-281

OracleParameter(string, OracleDbType, object, ParameterDirection)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, value, and direction.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType type, object obj,
ParameterDirection direction);
```

Parameters

- *parameterName*

  Specifies the parameter name.

- *type*

  Specifies the datatype of the `OracleParameter`.

- *obj*

  Specifies the value of the `OracleParameter`.

- *direction*

  Specifies one of the `ParameterDirection` values.

Remarks

Changing the `DbType` implicitly changes the `OracleDbType`.

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType` - `String`
- `ParameterDirection` - `Input`
- `isNullable` - `true`
- `offset` - `0`
- `OracleDbType` - `Varchar2`
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363
- "OracleParameterCollection Class" on page 4-281

OracleParameter(string, OracleDbType, int)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, and size.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType type, int size);
```

Parameters

- *parameterName*

  Specifies the parameter name.

- *type*

  Specifies the datatype of the `OracleParameter`.

- *size*

  Specifies the size of the `OracleParameter` value.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType - String`
- `ParameterDirection - Input`
- `isNullable - true`
- `offset - 0`
- `OracleDbType - Varchar2`

- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363
- "OracleParameterCollection Class" on page 4-281

OracleParameter(string, OracleDbType, int, string)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, size, and source column.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType type, int size,
 string srcColumn);
```

Parameters

- *`parameterName`*

  Specifies the parameter name.

- *`type`*

  Specifies the datatype of the `OracleParameter`.

- *`size`*

  Specifies the size of the `OracleParameter` value.

- *srcColumn*

  Specifies the name of the source column.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- DbType - String
- ParameterDirection - Input
- isNullable - true
- offset - 0
- OracleDbType - Varchar2
- ParameterAlias - Empty string
- ParameterName - Empty string
- Precision - 0
- Size - 0
- SourceColumn - Empty string
- SourceVersion - Current
- ArrayBindStatus - Success
- Value - null

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363

OracleParameter(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, datatype, size, direction, null indicator, precision, scale, source column, source version and parameter value.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType oraType, int size,
 ParameterDirection direction, bool isNullable, byte precision, byte scale,
 string srcColumn, DataRowVersion srcVersion, object obj);
```

Parameters

- *parameterName*

  Specifies the parameter name.

- *oraType*

  Specifies the datatype of the `OracleParameter`.

- *size*

  Specifies the size of the `OracleParameter` value.

- *direction*

  Specifies `ParameterDirection` value.

- *isNullable*

  Specifies if the parameter value can be `null`.

- *precision*

  Specifies the precision of the parameter value.

- *scale*

  Specifies the scale of the parameter value.

- *srcColumn*

  Specifies the name of the source column.

- *srcVersion*

  Specifies one of the `DataRowVersion` values.

- *obj*

  Specifies the parameter value.

Exceptions

`ArgumentException` - The supplied value does not belong to the type of `Value` property in any of the `OracleTypes`.

Remarks

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType` - `String`
- `ParameterDirection` - `Input`
- `isNullable` - `true`
- `offset` - `0`
- `OracleDbType` - `Varchar2`
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363

OracleParameter(string, OracleDbType, int, object, ParameterDirection)

This constructor instantiates a new instance of the `OracleParameter` class using the supplied parameter name, datatype, size, value, and direction.

Declaration

```
// C#
public OracleParameter(string parameterName, OracleDbType type, int size, object
obj, ParameterDirection direction);
```

Parameters

- *parameterName*

  Specifies the parameter name.

- *type*

  Specifies the datatype of the `OracleParameter`.

- *size*

  Specifies the size of the `OracleParameter` value.

- *obj*

  Specifies the value of the `OracleParameter`.

- *direction*

  Specifies one of the `ParameterDirection` values.

Remarks

Changing the `DbType` implicitly changes the `OracleDbType`.

Unless explicitly set in the constructor, all the properties have the default values.

Default Values:

- `DbType` - `String`
- `ParameterDirection` - `Input`
- `isNullable` - `true`
- `offset` - `0`
- `OracleDbType` - `Varchar2`
- `ParameterAlias` - Empty string
- `ParameterName` - Empty string
- `Precision` - `0`
- `Size` - `0`
- `SourceColumn` - Empty string
- `SourceVersion` - `Current`
- `ArrayBindStatus` - `Success`
- `Value` - `null`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-363
- "OracleParameterCollection Class" on page 4-281

OracleParameter Static Methods

`OracleParameter` static methods are listed in Table 4–89.

Table 4–89 OracleParameter Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

OracleParameter Properties

`OracleParameter` properties are listed in Table 4–90.

Table 4–90 OracleParameter Properties

Name	Description
ArrayBindSize	Specifies the input or output size of elements in `Value` property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
ArrayBindStatus	Specifies the input or output status of elements in `Value` property of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution
DbType	Specifies the datatype of the parameter using the `Data.DbType` enumeration type

Table 4–90 OracleParameter Properties (Cont.)

Name	Description
Direction	Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter
`IsNullable`	*This method is a no-op*
Offset	Specifies the offset to the `Value` property or offset to the elements in the `Value` property
OracleDbType	Specifies the Oracle datatype
ParameterName	Specifies the name of the parameter
Precision	Specifies the maximum number of digits used to represent the `Value` property
Scale	Specifies the number of decimal places to which `Value` property is resolved
Size	Specifies the maximum size, in bytes or characters, of the data transmitted to or from the server. For PL/SQL Associative Array Bind, `Size` specifies the maximum number of elements in PL/SQL Associative Array
SourceColumn	Specifies the name of the `DataTable` Column of the `DataSet`
SourceVersion	Specifies the `DataRowVersion` value to use when loading the `Value` property of the parameter
Status	Indicates the status of the execution related to the data in the `Value` property
Value	Specifies the value of the `Parameter`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

ArrayBindSize

This property specifies the input or output size of elements of `Value` property before or after an Array Bind or PL/SQL Associative Array execution.

Declaration

```
// C#
```

```
public int[] ArrayBindSize {get; set; }
```

Property Value

An array of `int` values specifying the size.

Remarks

Default = `null`.

This property is only used for variable size element types for an Array Bind or PL/SQL Associative Array. For fixed size element types, this property is ignored.

Each element in the `ArrayBindSize` corresponds to the bind size of an element in the `Value` property. Before execution, `ArrayBindSize` specifies the maximum size of each element to be bound in the `Value` property. After execution, it contains the size of each element returned in the `Value` property.

For binding a PL/SQL Associative Array, whose elements are of a variable-length element type, as an `InputOutput`, `Out`, or `ReturnValue` parameter, this property must be set, and the number of elements in `ArrayBindSize` must be equal to or greater than the number of elements in `Value` property.

Example

```
// C#
...
OracleParameter Param = new OracleParameter("name", OracleDbType.Varchar2);
Param.ArrayBindSize = new Int32[3];

// These sizes indicate the maximum size of the elements in parameter Value
// property.
Param.ArrayBindSize[0] = 100;
Param.ArrayBindSize[1] = 300;
Param.ArrayBindSize[2] = 200;
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "ArrayBindCount" on page 4-14
- "Size" on page 4-271 and "Value" on page 4-276 for more information on binding Associative Arrays
- "ArrayBindStatus" on page 4-264

ArrayBindStatus

This property specifies the input or output status of each element in the `Value` property before or after an Array Bind or PL/SQL Associative Array execution.

Declaration

```
// C#
public OracleParameterStatus[] ArrayBindStatus { get; set; }
```

Property Value

An array of `OracleParameterStatus` enumerated values.

Exceptions

`ArgumentOutofRangeException` - The `Status` value specified is invalid.

Remarks

Default = `null`.

`ArrayBindStatus` is used for Array Bind and PL/SQL Associative Array execution only.

Before execution, `ArrayBindStatus` indicates the bind status of each element in the `Value` property. After execution, it contains the execution status of each element in the `Value` property.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "ArrayBindCount" on page 4-14
- "OracleParameterStatus Enumeration" on page 4-363
- "Value" on page 4-276 for more information on binding Associative Arrays
- "ArrayBindSize" on page 4-262

CollectionType

This property specifies whether the `OracleParameter` represents a collection, and if so, specifies the collection type.

Declaration

```
// C#
public OracleCollectionType CollectionType { get; set; }
```

Property Value

An `OracleCollectionType` enumerated value.

Exceptions

`ArgumentException` - The `OracleCollectionType` value specified is invalid.

Remarks

Default = `OracleCollectionType.None`.

If `OracleParameter` is used to bind a PL/SQL Associative Array, then `CollectionType` must be set to `OracleCollectionType.PLSQLAssociativeArray`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

DbType

This property specifies the datatype of the parameter using the `Data.DbType` enumeration type.

Declaration

```
// C#
public DbType DbType {get; set; }
```

Property Value

A `DbType` enumerated value.

Implements

`IDataParameter`

Exceptions

`ArgumentException` - The `DbType` value specified is invalid.

Remarks

Default = `DbType.String`

`DbType` is the datatype of each element in the array if the `OracleParameter` object is used for Array Bind or PL/SQL Associative Array Bind execution.

Due to the link between `DbType` and `OracleDbType` properties, if the `DbType` property is set, the `OracleDbType` property is inferred from `DbType`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "Inference of OracleDbType from DbType" on page 3-23
- "CollectionType" on page 4-265

Direction

This property specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter.

Declaration

```
// C#
public ParameterDirection Direction { get; set; }
```

Property Value

A `ParameterDirection` enumerated value.

Implements

`IDataParameter`

Exceptions

`ArgumentOutOfRangeException` - The `ParameterDirection` value specified is invalid.

Remarks

Default = `ParameterDirection.Input`

Possible values: `Input`, `InputOutput`, `Output`, and `ReturnValue`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

Offset

This property specifies the offset to the `Value` property.

Declaration

```
// C#
public int Offset { get; set; }
```

Property Value

An `int` that specifies the offset.

Exceptions

`ArgumentOutOfRangeException` - The `Offset` value specified is invalid.

Remarks

Default = `0`

For Array Bind and PL/SQL Associative Array Bind, `Offset` applies to every element in the `Value` property.

The `Offset` property is used for binary and string data types. The `Offset` property represents the number of bytes for binary types and the number of characters for strings. The count for strings does not include the terminating character if a `null` is referenced. The `Offset` property is used by parameters of the following types:

- `OracleDbType.BFile`
- `OracleDbType.Blob`
- `OracleDbType.LongRaw`
- `OracleDbType.Raw`
- `OracleDbType.Char`
- `OracleDbType.Clob`
- `OracleDbType.NClob`
- `OracleDbType.NChar`
- `OracleDbType.NVarchar2`
- `OracleDbType.Varchar2`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

OracleDbType

This property specifies the Oracle datatype.

Declaration

```
// C#
public OracleDbType OracleDbType { get; set; }
```

Property Value

An `OracleDbType` enumerated value.

Remarks

Default = `OracleDbType.Varchar2`

If the `OracleParameter` object is used for Array Bind or PL/SQL Associative Array Bind execution, `OracleDbType` is the datatype of each element in the array.

The `OracleDbType` property and `DbType` property are linked. Therefore, setting the `OracleDbType` property changes the `DbType` property to a supporting `DbType`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleDbType Enumeration" on page 4-361
- "Inference of DbType from OracleDbType" on page 3-22
- "CollectionType" on page 4-265

ParameterName

This property specifies the name of the parameter.

Declaration

```
// C#
public string ParameterName { get; set; }
```

Property Value

`String`

Implements

`IDataParameter`

Remarks

Default = `null`

Oracle supports `ParameterName` up to 30 characters.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

Precision

This property specifies the maximum number of digits used to represent the `Value` property.

Declaration

```
// C#
Public byte Precision { get; set; }
```

Property Value

`byte`

Remarks

Default = `0`

The `Precision` property is used by parameters of type `OracleDbType.Decimal`.

Oracle supports `Precision` range from `0` to `38`.

For Array Bind and PL/SQL Associative Array Bind, `Precision` applies to each element in the `Value` property.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- "Value" on page 4-276
- OracleParameter Members

Scale

This property specifies the number of decimal places to which `Value` property is resolved.

Declaration

```
// C#
public byte Scale { get; set; }
```

Property Value

`byte`

Remarks

Default = `0`.

`Scale` is used by parameters of type `OracleDbType.Decimal`.

Oracle supports `Scale` between `-84` and `127`.

For Array Bind and PL/SQL Associative Array Bind, `Scale` applies to each element in the `Value` property.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "Value" on page 4-276

Size

This property specifies the maximum size, in bytes or characters, of the data transmitted to or from the server.

For PL/SQL Associative Array Bind, `Size` specifies the maximum number of elements in PL/SQL Associative Array.

Declaration

```
// C#
public int Size { get; set;}
```

Property Value

`int`

Exceptions

`ArgumentOutOfRangeException` - The `Size` value specified is invalid.

`InvalidOperationException` - The `Size` = 0 when the `OracleParameter` object is used to bind a PL/SQL Associative Array.

Remarks

The default value is `0`.

Before execution, this property specifies the maximum size to be bound in the `Value` property. After execution, it contains the size of the type in the `Value` property.

`Size` is used for parameters of the following types:

- `OracleDbType.Blob`
- `OracleDbType.Char`
- `OracleDbType.Clob`
- `OracleDbType.LongRaw`
- `OracleDbType.NChar`
- `OracleDbType.NClob`
- `OracleDbType.NVarchar2`
- `OracleDbType.Raw`
- `OracleDbType.Varchar2`

The value of `Size` is handled as follows:

- Fixed length datatypes: ignored
- Variable length datatypes: describes the maximum amount of data transmitted to or from the server. For character data, `Size` is in number of characters and for binary data, it is in number of bytes.

If the `Size` is not explicitly set, it is inferred from the actual size of the specified parameter value when binding.

Note: `Size` does not include the null terminating character for the string data.

If the `OracleParameter` object is used to bind a PL/SQL Associative Array, `Size` specifies the maximum number of elements in the PL/SQL Associative Array. Before the execution, this property specifies the maximum number of elements in

the PL/SQL Associative Array. After the execution, it specifies the current number of elements returned in the PL/SQL Associative Array. For `Output` and `InputOutput` parameters and return values, `Size` specifies the maximum number of elements in the PL/SQL Associative Array.

ODP.NET does not support binding an empty PL/SQL Associative Array. Therefore, `Size` cannot be set to `0` when the `OracleParameter` object is used to bind a PL/SQL Associative Array.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "OracleDbType Enumeration" on page 4-361
- "CollectionType" on page 4-265
- "ArrayBindSize" on page 4-262
- "ArrayBindStatus" on page 4-264
- "Value" on page 4-276

SourceColumn

This property specifies the name of the `DataTable` Column of the `DataSet`.

Declaration

```
// C#
public string SourceColumn { get; set; }
```

Property Value

A `string`.

Implements

`IDataParameter`

Remarks

Default = empty string

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

SourceVersion

This property specifies the `DataRowVersion` value to use when loading the `Value` property of the parameter.

Declaration

```
// C#
public DataRowVersion SourceVersion { get; set; }
```

Property Value

`DataRowVersion`

Implements

`IDataParameter`

Exceptions

`ArgumentOutOfRangeException` - The `DataRowVersion` value specified is invalid.

Remarks

Default = `DataRowVersion.Current`

`SourceVersion` is used by the `OracleDataAdapter.UpdateCommand()` during the `OracleDataAdapter.Update` to determine whether the original or current value is used for a parameter value. This allows primary keys to be updated. This property is ignored by the `OracleDataAdapter.InsertCommand()` and the `OracleDataAdapter.DeleteCommand()`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

Status

This property indicates the status of the execution related to the data in the `Value` property.

Declaration

```
// C#
public OracleParameterStatus Status { get; set; }
```

Property Value

An `OracleParameterStatus` enumerated value.

Exceptions

`ArgumentOutOfRangeException` - The `Status` value specified is invalid.

Remarks

Default = `OracleParameterStatus.Success`

Before execution, this property indicates the bind status related to the `Value` property. After execution, it returns the status of the execution.

`Status` indicates whether:

- A `NULL` is fetched from a column.
- Truncation has occurred during the fetch; then `Value` was not big enough to hold the data.
- A `NULL` is to be inserted into a database column; then `Value` is ignored, and a `NULL` is inserted into a database column.

  This property is ignored for Array Bind and PL/SQL Associative Array Bind. Instead, `ArrayBindStatus` property is used.

  See Also:

  - Oracle.DataAccess.Client Namespace
  - OracleParameter Class
  - OracleParameter Members
  - "OracleParameterStatus Enumeration" on page 4-363
  - "ArrayBindStatus" on page 4-264

Value

This property specifies the value of the `Parameter`.

Declaration

```
// C#
public object Value { get; set; }
```

Property Value

An `object`.

Implements

`IDataParameter`

Exceptions

`ArgumentException` - The `Value` property specified is invalid.

`InvalidArgumentException`- The `Value` property specified is invalid.

Remarks

Default = `null`

If the `OracleParameter` object is used for Array Bind or PL/SQL Associative Array, `Value` is an array of parameter values.

The `Value` property can be overwritten by `OracleDataAdapter.Update()`.

The provider attempts to convert any type of value if it supports the `IConvertible` interface. Conversion errors occur if the specified type is not compatible with the value.

When sending a `null` parameter value to the database, the user must specify `DBNull`, not `null`. The `null` value in the system is an empty object that has no value. `DBNull` is used to represent `null` values. The user can also specify a `null` value by setting `Status` to `OracleParameterStatus.NullValue`. In this case, the provider sends a `null` value to the database.

If neither `OracleDbType` nor `DbType` are set, their values can be inferred by `Value`.

See Also:

- Tables in section "Inference of DbType and OracleDbType from Value" on page 3-24
- "ArrayBindCount" on page 4-14
- "ArrayBindSize" on page 4-262
- "ArrayBindStatus" on page 4-264
- "OracleDbType Enumeration" on page 4-361

For input parameters the value is:

- Bound to the `OracleCommand` that is sent to the server.
- Converted to the datatype specified in `OracleDbType` or `DbType` when the provider sends the data to the server.

For output parameters the value is:

- Set on completion of the `OracleCommand` (true for return value parameters also).
- Set to the data from the server, to the datatype specified in `OracleDbType` or `DbType`.

When array binding is used with:

- Input parameter - `Value` should be set to an array of values. `OracleCommand.ArrayBindCount` should be set to a value that is greater than zero to indicate the number of elements to be bound.

  The number of elements in the array should be equal to the `OracleCommand.ArrayBindCount` property; otherwise, their minimum value is used to bind the elements in the array.
- Output parameter - `OracleCommand.ArrayBindCount` should be set to a value that is greater than zero to indicate the number of elements to be retrieved (for `SELECT` statements).

When PL/SQL Associative Array binding is used with:

- Input parameter – Value should be set to an array of values. `CollectionType` should be set to `OracleCollection.PLSQLAssociativeArray.Size` should be set to specify the possible maximum number of array elements in the PL/SQL Associative Array. If `Size` is smaller than the number of elements in

`Value`, then `Size` specifies the number of elements in the Value property to be bound.

- Output parameter - `CollectionType` should be set to `OracleCollection.PLSQLAssociativeArray`. `Size` should be set to specify the maximum number of array elements in PL/SQL Associative Array.

Each parameter should have a value. To bind a parameter with a `null` value, set `Value` to `DBNull.Value`, or set `Status` to `OracleParameterStatus.NullInsert`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members
- "ArrayBindCount" on page 4-14
- "OracleParameterStatus Enumeration" on page 4-363

OracleParameter Public Methods

`OracleParameter` public methods are listed in Table 4–91.

Table 4–91 OracleParameter Public Methods

Public Method	Description
Clone	Creates a shallow copy of an `OracleParameter` object
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Releases allocated resources
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
`ToString`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

Clone

This method creates a shallow copy of an `OracleParameter` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleParameter` object.

Implements

`ICloneable`

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
//Need a proper casting for the return value when cloned
OracleParameter param_cloned = (OracleParameter) param.Clone();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

Dispose

This method releases resources allocated for an `OracleParameter` object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameter Class
- OracleParameter Members

OracleParameterCollection Class

An `OracleParameterCollection` class represents a collection of all parameters relevant to an `OracleCommand` object and their mappings to `DataSet` columns.

Class Inheritance

```
Object
  MarshalByRefObject
    OracleParameterCollection
```

Declaration

```
// C#
public sealed class OracleParameterCollection : MarshalByRefObject,
  IDataParameterCollection, IList, ICollection, IEnumerable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The position of an `OracleParameter` added into the `OracleParameterCollection` is the binding position in the SQL statement. Position is `0`-based and is used only for positional binding. If named binding is used, the position of an `OracleParameter` in the `OracleParameterCollection` is ignored.

Example

```
// C#
string conStr = "User Id=scott;Password=tiger;Data Source=oracle";

// Create the OracleConnection
OracleConnection con = new OracleConnection(conStr);
con.Open();

// Create the OracleCommand
OracleCommand cmd = new OracleCommand();
cmd.Connection = con;

// Create OracleParameter
```

```
OracleParameter [] prm = new OracleParameter[3];

// Bind parameters
prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal, 1234,
    ParameterDirection.Input);
prm[1] = cmd.Parameters.Add("paramEname", OracleDbType.Varchar2,
    "Client", ParameterDirection.Input);
prm[2] = cmd.Parameters.Add("paramDeptNo", OracleDbType.Decimal,
    10, ParameterDirection.Input);

cmd.CommandText = "insert into emp(empno, ename, deptno) values(:1, :2, :3)";
cmd.ExecuteNonQuery();

// Remove OracleParameter objects from the collection
cmd.Parameters.Clear();

// Dispose OracleCommand object
cmd.Dispose();

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Members
- OracleParameterCollection Static Methods
- OracleParameterCollection Properties
- OracleParameterCollection Public Methods

OracleParameterCollection Members

`OracleParameterCollection` members are listed in the following tables:

OracleParameterCollection Static Methods

`OracleParameterCollection` static methods are listed in Table 4–92.

Table 4–92 OracleParameterCollection Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleParameterCollection Properties

`OracleParameterCollection` properties are listed in Table 4–93.

Table 4–93 OracleParameterCollection Properties

Name	Description
Count	Specifies the number of `OracleParameters` in the collection
Item	Gets and sets the `OracleParameter` object (Overloaded)

OracleParameterCollection Public Methods

`OracleParameterCollection` public methods are listed in Table 4–94.

Table 4–94 OracleParameterCollection Public Methods

Public Method	Description
Add	Adds objects to the collection (Overloaded)
Clear	Removes all the `OracleParameter` objects from the collection
Contains	Indicates whether objects exist in the collection (Overloaded)
CopyTo	Copies `OracleParameter` objects from the collection, starting with the supplied `index` to the supplied `array`
`CreateObjRef`	Inherited from `MarshalByRefObject`
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`

Table 4–94 OracleParameterCollection Public Methods (Cont.)

Public Method	Description
GetType	Inherited from Object
InitializeLifetimeService	Inherited from MarshalByRefObject
IndexOf	Returns the index of the objects in the collection (Overloaded)
Insert	Inserts the supplied OracleParameter to the collection at the specified index
Remove	Removes objects from the collection
RemoveAt	Removes objects from the collection by location (Overloaded)
ToString	Inherited from Object

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class

OracleParameterCollection Static Methods

OracleParameterCollection static methods are listed in Table 4–95.

Table 4–95 OracleParameterCollection Static Methods

Methods	Description
Equals	Inherited from Object (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

OracleParameterCollection Properties

OracleParameterCollection properties are listed in Table 4–96.

Table 4–96 OracleParameterCollection Properties

Name	Description
Count	Specifies the number of `OracleParameters` in the collection
Item	Gets and sets the `OracleParameter` object (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Count

This property specifies the number of `OracleParameter` objects in the collection.

Declaration

```
// C#
public int Count {get;}
```

Property Value

The number of `OracleParameter` objects.

Implements

`ICollection`

Remarks

Default = `0`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Item

`Item` gets and sets the `OracleParameter` object.

Overload List:

- Item[int]

  This property gets and sets the `OracleParameter` object at the index specified by the supplied `parameterIndex`.

- Item[string]

  This property gets and sets the `OracleParameter` object using the parameter name specified by the supplied `parameterName`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Item[int]

This property gets and sets the `OracleParameter` object at the index specified by the supplied `parameterIndex`.

Declaration

```
// C#
public object Item[int parameterIndex] {get; set;}
```

Property Value

An object.

Implements

`IList`

Exceptions

`IndexOutOfRangeException` - The supplied index does not exist.

Remarks

The `OracleParameterCollection` class is a zero-based index.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Item[string]

This property gets and sets the `OracleParameter` object using the parameter name specified by the supplied `parameterName`.

Declaration

```
// C#
public OracleParameter Item[string parameterName] {get; set;};
```

Property Value

An `OracleParameter`.

Implements

`IDataParameterCollection`

Exceptions

`IndexOutOfRangeException` - The supplied parameter name does not exist.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

OracleParameterCollection Public Methods

`OracleParameterCollection` public methods are listed in Table 4–97.

Table 4–97 OracleParameterCollection Public Methods

Public Method	Description
Add	Adds objects to the collection (Overloaded)
Clear	Removes all the `OracleParameter` objects from the collection

Table 4–97 OracleParameterCollection Public Methods (Cont.)

Public Method	Description
Contains	Indicates whether objects exist in the collection (Overloaded)
CopyTo	Copies `OracleParameter` objects from the collection, starting with the supplied `index` to the supplied `array`
`CreateObjRef`	Inherited from `MarshalByRefObject`
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
IndexOf	Returns the `index` of the objects in the collection (Overloaded)
Insert	Inserts the supplied `OracleParameter` to the collection at the specified `index`
Remove	Removes objects from the collection
RemoveAt	Removes objects from the collection by location (Overloaded)
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Add

`Add` adds objects to the collection.

Overload List:

- Add(object)

  This method adds the supplied object to the collection.

- Add(OracleParameter)

  This method adds the supplied `OracleParameter` object to the collection.

- Add(string, object)

  This method adds an `OracleParameter` object to the collection using the supplied name and object value.

- Add(string, OracleDbType)

  This method adds an `OracleParameter` object to the collection using the supplied name and database type.

- Add(string, OracleDbType, ParameterDirection)

  This method adds an `OracleParameter` object to the collection using the supplied name, database type, and direction.

- Add(string, OracleDbType, object, ParameterDirection)

  This method adds an `OracleParameter` object to the collection using the supplied name, database type, parameter value, and direction.

- Add(string, OracleDbType, int, object, ParameterDirection)

  This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, parameter value, and direction.

- Add(string, OracleDbType, int)

  This method adds an `OracleParameter` object to the collection using the supplied name, database type, and size.

- Add (string, OracleDbType, int, string)

  This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, and source column.

- Add(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)

  This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, direction, null indicator, precision, scale, source column, source version, and parameter value.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Add(object)

This method adds the supplied object to the collection.

Declaration

```
// C#
public int Add(object obj);
```

Parameters

- *obj*

  Specifies the supplied object.

Return Value

The index at which the new `OracleParameter` is added.

Implements

`IList`

Remarks

`InvalidCastException` - The supplied *obj* cannot be cast to an `OracleParameter` object.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Add(OracleParameter)

This method adds the supplied `OracleParameter` object to the collection.

Declaration

```
// C#
public OracleParameter Add(OracleParameter paramObj);
```

Parameters

- *paramObj*

  Specifies the supplied `OracleParameter` object.

Return Value

The newly created `OracleParameter` object which was added to the collection.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Add(string, object)

This method adds an `OracleParameter` object to the collection using the supplied name and object value

Declaration

```
// C#
public OracleParameter Add(string name, object val);
```

Parameters

- *name*

  Specifies the parameter name.

- *val*

  Specifies the `OracleParameter` value.

Return Value

The newly created `OracleParameter` object which was added to the collection.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Add(string, OracleDbType)

This method adds an `OracleParameter` object to the collection using the supplied name and database type.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType);
```

Parameters

- *name*

  Specifies the parameter name.

- *dbType*

  Specifies the datatype of the `OracleParameter`.

Return Value

The newly created `OracleParameter` object which was added to the collection.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Add(string, OracleDbType, ParameterDirection)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, and direction.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, ParameterDirection
direction);
```

Parameters

- *name*

  Specifies the parameter name.

- *dbType*

  Specifies the datatype of the `OracleParameter`.

- *direction*

  Specifies the `OracleParameter` direction.

Return Value

The newly created `OracleParameter` object which was added to the collection.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members
- "OracleDbType Enumeration" on page 4-361

Add(string, OracleDbType, object, ParameterDirection)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, parameter value, and direction.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, object val,
 ParameterDirection dir);
```

Parameters

- *name*

  Specifies the parameter name.

- *dbType*

  Specifies the datatype of the `OracleParameter`.

- *val*

  Specifies the `OracleParameter` value.

- *dir*

  Specifies one of the ParameterDirection values.

Return Value

The newly created OracleParameter object which was added to the collection.

Example

```
// C#
...
OracleParameter prm = new OracleParameter();
prm = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal, 1234,
   ParameterDirection.Input);

cmd.CommandText = "insert into NumTable(numcol) values(:1)";
cmd.ExecuteNonQuery();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members
- "OracleDbType Enumeration" on page 4-361

Add(string, OracleDbType, int, object, ParameterDirection)

This method adds an OracleParameter object to the collection using the supplied name, database type, size, parameter value, and direction.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size,
   object val, ParameterDirection dir;
```

Parameters

- *name*

  Specifies the parameter name.

- *dbType*

Specifies the datatype of the `OracleParameter`.

- *`size`*

  Specifies the size of `OracleParameter`.

- *`val`*

  Specifies the `OracleParameter` value.

- *`dir`*

  Specifies one of the `ParameterDirection` values.

Return Value

The newly created `OracleParameter` object which was added to the collection.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members
- "OracleDbType Enumeration" on page 4-361

Add(string, OracleDbType, int)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, and size.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size);
```

Parameters

- *`name`*

  Specifies the parameter name.

- *`dbType`*

  Specifies the datatype of the `OracleParameter`.

- *`size`*

  Specifies the size of `OracleParameter`.

Return Value

The newly created `OracleParameter` object which was added to the collection.

Example

```
// C#
...
OracleParameter prm = new OracleParameter();
prm = cmd.Parameters.Add("param1", OracleDbType.Decimal, 10);
prm.Direction = ParameterDirection.Input;
prm.Value = 1111;

cmd.CommandText = "insert into NumTable(numcol) values(:1)";
cmd.ExecuteNonQuery();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Add (string, OracleDbType, int, string)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, and source column.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size,
  string srcColumn);
```

Parameters

- *name*

  Specifies the parameter name.

- *dbType*

  Specifies the datatype of the `OracleParameter`.

- *size*

  Specifies the size of `OracleParameter`.

- *srcColumn*

  Specifies the name of the source column.

Return Value

An `OracleParameter`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Add(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)

This method adds an `OracleParameter` object to the collection using the supplied name, database type, size, direction, null indicator, precision, scale, source column, source version, and parameter value.

Declaration

```
// C#
public OracleParameter Add(string name, OracleDbType dbType, int size,
  ParameterDirection dir, bool isNullable, byte precision,
  byte scale, string srcColumn, DataRowVersion version, object val);
```

Parameters

- *name*

  Specifies the parameter name.

- *dbType*

  Specifies the datatype of the `OracleParameter`.

- *size*

  Specifies the size of `OracleParameter`.

- *dir*

  Specifies one of the `ParameterDirection` values.

- *isNullable*

  Specifies if the `parameter` value can be `null`.

- *precision*

  Specifies the precision of the `parameter` value.

- *scale*

  Specifies the scale of the `parameter` value.

- *srcColumn*

  Specifies the name of the source column.

- *version*

  Specifies one of the `DataRowVersion` values.

- *val*

  Specifies the `parameter` value.

Return Value

The newly created `OracleParameter` object which was added to the collection.

Exceptions

`ArgumentException` - The type of supplied *val* does not belong to the type of `Value` property in any of the ODP.NET Types.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Clear

This method removes all the `OracleParameter` objects from the collection.

Declaration

```
// C#
public void Clear();
```

Implements

`IList`

Example

```
// C#
...
OracleParameter [] prm = new OracleParameter[3];

prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal,
   1234, ParameterDirection.Input);
prm[1] = cmd.Parameters.Add("paramEname", OracleDbType.Varchar2,
   "Client", ParameterDirection.Input);
prm[2] = cmd.Parameters.Add("paramDeptNo", OracleDbType.Decimal, 10,
    ParameterDirection.Input);

cmd.CommandText = "insert into emp(empno, ename, deptno) values(:1,:2, :3)";
cmd.ExecuteNonQuery();

// This method removes all the parameters from the parameter collection.
cmd.Parameters.Clear();
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Contains

`Contains` indicates whether the supplied object exists in the collection.

Overload List:

- Contains(object)

  This method indicates whether the supplied object exists in the collection.

- Contains(string)

  This method indicates whether an `OracleParameter` object exists in the collection using the supplied string.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Contains(object)

This method indicates whether the supplied object exists in the collection.

Declaration

```
// C#
public bool Contains(object obj)
```

Parameters

- *obj*

  Specifies the object.

Return Value

A `bool` that indicates whether or not the `OracleParameter` specified is inside the collection.

Implements

`IList`

Exceptions

`InvalidCastException` - The supplied *obj* is not an `OracleParameter` object.

Remarks

Returns `true` if the collection contains the `OracleParameter` object; otherwise, returns `false`.

Example

```
...
prm = cmd.Parameters.Add("param1", OracleDbType.Decimal, 1234,
     ParameterDirection.Input);
if (cmd.Parameters.Contains((Object)prm))
// This method removes a particular parameter from the parameter collection.
cmd.Parameters.Remove((Object) prm);
```

```
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Contains(string)

This method indicates whether an `OracleParameter` object exists in the collection using the supplied string.

Declaration

```
// C#
public bool Contains(string name);
```

Parameters

- *name*

  Specifies the name of `OracleParameter` object.

Return Value

Returns `true` if the collection contains the `OracleParameter` object with the specified parameter name; otherwise, returns `false`.

Implements

`IDataParameterCollection`

Example

```
...
prm = cmd.Parameters.Add("param1", OracleDbType.Decimal, 1234, +
     ParameterDirection.Input);
if (cmd.Parameters.Contains((Object)prm))
// This method removes a particular parameter from the parameter collection.
cmd.Parameters.Remove((Object) prm);
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

CopyTo

This method copies `OracleParameter` objects from the collection, starting with the supplied `index` to the supplied `array`.

Declaration

```
// C#
public void CopyTo(Array array, int index);
```

Parameters

- *`array`*

  Specifies the array.

- *`index`*

  Specific the index to array.

Implements

`ICollection`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

IndexOf

`IndexOf` returns the index of the `OracleParameter` object in the collection.

Overload List:

- IndexOf(object)

  This method returns the index of the `OracleParameter` object in the collection.

- IndexOf(String)

  This method returns the `index` of the `OracleParameter` object with the specified name in the collection.

  See Also:

  - Oracle.DataAccess.Client Namespace
  - OracleParameterCollection Class
  - OracleParameterCollection Members

IndexOf(object)

This method returns the index of the `OracleParameter` object in the collection.

Declaration

```
// C#
public int IndexOf(object obj);
```

Parameters

- *obj*

  Specifies the object.

Return Value

Returns the index of the `OracleParameter` object in the collection.

Implements

`IList`

Exceptions

`InvalidCastException` - The supplied *obj* cannot be cast to an `OracleParameter` object.

Remarks

Returns the `index` of the supplied `OracleParameter` *obj* in the collection.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

IndexOf(String)

This method returns the `index` of the `OracleParameter` object with the specified name in the collection.

Declaration

```
// C#
public int IndexOf(String name);
```

Parameters

- *name*

  Specifies the name of parameter.

Return Value

Returns the `index` of the supplied `OracleParameter` in the collection.

Implements

`IDataParameterCollection`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Insert

This method inserts the supplied `OracleParameter` object to the collection at the specified `index`.

Declaration

```
// C#
public void Insert(int index, object obj);
```

Parameters

- *index*

  Specifies the index.

- *obj*

  Specifies the `OracleParameter` object.

Implements

`IList`

Remarks

An `InvalidCastException` is thrown if the supplied *obj* cannot be cast to an `OracleParameter` object.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

Remove

This method removes the supplied `OracleParameter` from the collection.

Declaration

```
// C#
public void Remove(object obj);
```

Parameters

- *obj*

  Specifies the object to remove.

Implements

`IList`

Exceptions

`InvalidCastException` - The supplied *obj* cannot be cast to an `OracleParameter` object.

Example

```
...
prm = cmd.Parameters.Add("param1", OracleDbType.Decimal, 1234,
     ParameterDirection.Input);
if (cmd.Parameters.Contains((Object)prm))
// This method removes a particular parameter from the parameter collection.
cmd.Parameters.Remove((Object) prm);
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

RemoveAt

`RemoveAt` removes the `OracleParameter` object from the collection by location.

Overload List:

- RemoveAt(int)

  This method removes from the collection the `OracleParameter` object located at the index specified by the supplied index.

- RemoveAt(String)

  This method removes from the collection the `OracleParameter` object specified by the supplied name.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

RemoveAt(int)

This method removes from the collection the `OracleParameter` object located at the index specified by the supplied index.

Declaration

```
// C#
public void RemoveAt(int index);
```

Parameters

- *index*

  Specifies the index from which the `OracleParameter` is to be removed.

Implements

`IList`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

RemoveAt(String)

This method removes from the collection the `OracleParameter` object specified by the supplied name.

Declaration

```
// C#
public void RemoveAt(String name);
```

Parameters

- *name*

  The name of the `OracleParameter` object to be removed from the collection.

Implements

`IDataParameterCollection`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleParameterCollection Class
- OracleParameterCollection Members

OracleRowUpdatedEventHandler Delegate

The `OracleRowUpdatedEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdated` event.

Declaration

```
// C#
public delegate void OracleRowUpdatedEventHandler(object sender,
    OracleRowUpdatedEventArgs eventArgs);
```

Parameters

- *sender*

  The source of the event.

- *eventArgs*

  The `OracleRowUpdatedEventArgs` object that contains the event data.

Remarks

Event callbacks can be registered through this event delegate for applications that wish to be notified after a row is updated.

In the .NET framework, the convention of an event delegate requires two parameters: the object that raises the event and the event data.

See Also:

- Oracle.DataAccess.Client Namespace
- "OracleDataAdapter Class" on page 4-86

OracleRowUpdatedEventArgs Class

The `OracleRowUpdatedEventArgs` class provides event data for the `OracleDataAdapter.RowUpdated` event.

Class Inheritance

```
Object
  EventArgs
    RowUpdatedEventArgs
      OracleRowUpdatedEventArgs
```

Declaration

```
// C#
public sealed class OracleRowUpdatedEventArgs : RowUpdatedEventArgs
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

The example for the `RowUpdated` event shows how to use `OracleRowUpdatedEventArgs`. See "Example" on page 4-108.

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatedEventArgs Members
- OracleRowUpdatedEventArgs Constructor
- OracleRowUpdatedEventArgs Static Methods
- OracleRowUpdatedEventArgs Properties
- OracleRowUpdatedEventArgs Public Methods
- OracleDataAdapter Class on page 4-86

OracleRowUpdatedEventArgs Members

`OracleRowUpdatedEventArgs` members are listed in the following tables:

OracleRowUpdatedEventArgs Constructors

`OracleRowUpdatedEventArgs` constructors are listed in Table 4–98.

Table 4–98 OracleRowUpdatedEventArgs Constructors

Constructor	Description
OracleRowUpdatedEventArgs Constructor	Instantiates a new instance of `OracleRowUpdatedEventArgs` class

OracleRowUpdatedEventArgs Static Methods

The `OracleRowUpdatedEventArgs` static methods are listed in Table 4–99.

Table 4–99 OracleRowUpdatedEventArgs Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleRowUpdatedEventArgs Properties

The `OracleRowUpdatedEventArgs` properties are listed in Table 4–100.

Table 4–100 OracleRowUpdatedEventArgs Properties

Name	Description
Command	Specifies the `OracleCommand` that is used when `OracleDataAdapter.Update()` is called
`Errors`	Inherited from `RowUpdatedEventArgs`
`RecordsAffected`	Inherited from `RowUpdatedEventArgs`
`Row`	Inherited from `RowUpdatedEventArgs`
`StatementType`	Inherited from `RowUpdatedEventArgs`
`Status`	Inherited from `RowUpdatedEventArgs`
`TableMapping`	Inherited from `RowUpdatedEventArgs`

OracleRowUpdatedEventArgs Public Methods

The `OracleRowUpdatedEventArgs` properties are listed in Table 4–101.

Table 4–101 OracleRowUpdatedEventArgs Public Methods

Name	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatedEventArgs Class

OracleRowUpdatedEventArgs Constructor

The `OracleRowUpdatedEventArgs` constructor creates a new `OracleRowUpdatedEventArgs` instance.

Declaration

```
// C#
public OracleRowUpdatedEventArgs(DataRow row,IDbCommand command,
   StatementType statementType, DataTableMapping tableMapping);
```

Parameters

- *row*

  The DataRow sent for Update.

- *command*

  The IDbCommand executed during the Update.

- *statementType*

  The StatementType Enumeration value indicating the type of SQL statement executed.

- *tableMapping*

  The DataTableMapping used for the Update.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members

OracleRowUpdatedEventArgs Static Methods

The OracleRowUpdatedEventArgs static methods are listed in Table 4–102.

Table 4–102 OracleRowUpdatedEventArgs Static Methods

Methods	Description
Equals	Inherited from Object (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members

OracleRowUpdatedEventArgs Properties

The OracleRowUpdatedEventArgs properties are listed in Table 4–103.

Table 4–103 OracleRowUpdatedEventArgs Properties

Name	Description
Command	Specifies the `OracleCommand` that is used when `OracleDataAdapter.Update()` is called
`Errors`	Inherited from `RowUpdatedEventArgs`
`RecordsAffected`	Inherited from `RowUpdatedEventArgs`
`Row`	Inherited from `RowUpdatedEventArgs`
`StatementType`	Inherited from `RowUpdatedEventArgs`
`Status`	Inherited from `RowUpdatedEventArgs`
`TableMapping`	Inherited from `RowUpdatedEventArgs`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members

Command

This property specifies the `OracleCommand` that is used when `OracleDataAdapter.Update()` is called.

Declaration

```
// C#
public new OracleCommand Command {get;}
```

Property Value

The `OracleCommand` executed when `Update` is called.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members

OracleRowUpdatedEventArgs Public Methods

The `OracleRowUpdatedEventArgs` properties are listed in Table 4–104.

Table 4–104 OracleRowUpdatedEventArgs Public Methods

Name	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members

OracleRowUpdatingEventArgs Class

The `OracleRowUpdatingEventArgs` class provides event data for the `OracleDataAdapter.RowUpdating` event.

Class Inheritance

```
Object
  EventArgs
    RowUpdatingEventArgs
      OracleRowUpdatingEventArgs
```

Declaration

```
// C#
public sealed class OracleRowUpdatingEventArgs : RowUpdatingEventArgs
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

The example for the `RowUpdated` event shows how to use `OracleRowUpdatingEventArgs`. See "Example" on page 4-108.

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatingEventArgs Members
- OracleRowUpdatingEventArgs Constructor
- OracleRowUpdatingEventArgs Static Methods
- OracleRowUpdatingEventArgs Properties
- OracleRowUpdatingEventArgs Public Methods
- "OracleDataAdapter Class" on page 4-86

OracleRowUpdatingEventArgs Members

`OracleRowUpdatingEventArgs` members are listed in the following tables:

OracleRowUpdatingEventArgs Constructors

`OracleRowUpdatingEventArgs` constructors are listed in Table 4–105.

Table 4–105 OracleRowUpdatingEventArgs Constructors

Constructor	Description
OracleRowUpdatingEventArgs Constructor	Instantiates a new instance of `OracleRowUpdatingEventArgs` class (Overloaded)

OracleRowUpdatingEventArgs Static Methods

The `OracleRowUpdatingEventArgs` static methods are listed in Table 4–106.

Table 4–106 OracleRowUpdatingEventArgs Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleRowUpdatingEventArgs Properties

The `OracleRowUpdatingEventArgs` properties are listed in Table 4–107.

Table 4–107 OracleRowUpdatingEventArgs Properties

Name	Description
Command	Specifies the `OracleCommand` that is used when the `OracleDataAdapter.Update()` is called
`Errors`	Inherited from `RowUpdatingEventArgs`
`Row`	Inherited from `RowUpdatingEventArgs`
`StatementType`	Inherited from `RowUpdatingEventArgs`
`Status`	Inherited from `RowUpdatingEventArgs`
`TableMapping`	Inherited from `RowUpdatingEventArgs`

OracleRowUpdatingEventArgs Public Methods

The `OracleRowUpdatingEventArgs` public methods are listed in Table 4–108.

Table 4–108 OracleRowUpdatingEventArgs Public Methods

Name	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatingEventArgs Class

OracleRowUpdatingEventArgs Constructor

The `OracleRowUpdatingEventArgs` constructor creates a new instance of the `OracleRowUpdatingEventArgs` class using the supplied data row, `IDbCommand`, type of SQL statement, and table mapping.

Declaration

```
// C#
public OracleRowUpdatingEventArgs(DataRow row, IDbCommand command,
   StatementType statementType, DataTableMapping tableMapping);
```

Parameters

- *row*

  The DataRow sent for Update.

- *command*

  The IDbCommand executed during the Update.

- *statementType*

  The StatementType enumeration value indicating the type of SQL statement executed.

- *tableMapping*

  The DataTableMapping used for the Update.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members

OracleRowUpdatingEventArgs Static Methods

The OracleRowUpdatingEventArgs static methods are listed in Table 4–109.

Table 4–109 OracleRowUpdatingEventArgs Static Methods

Methods	Description
Equals	Inherited from Object (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members

OracleRowUpdatingEventArgs Properties

The OracleRowUpdatingEventArgs properties are listed in Table 4–110.

Table 4–110 OracleRowUpdatingEventArgs Properties

Name	Description
Command	Specifies the `OracleCommand` that is used when the `OracleDataAdapter.Update()` is called
`Errors`	Inherited from `RowUpdatingEventArgs`
`Row`	Inherited from `RowUpdatingEventArgs`
`StatementType`	Inherited from `RowUpdatingEventArgs`
`Status`	Inherited from `RowUpdatingEventArgs`
`TableMapping`	Inherited from `RowUpdatingEventArgs`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members

Command

This property specifies the `OracleCommand` that is used when the `OracleDataAdapter.Update()` is called.

Declaration

```
// C#
public new OracleCommand Command {get; set;}
```

Property Value

The `OracleCommand` executed when `Update` is called.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members

OracleRowUpdatingEventArgs Public Methods

The `OracleRowUpdatingEventArgs` public methods are listed in Table 4–111.

Table 4–111 OracleRowUpdatingEventArgs Public Methods

Name	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members

OracleRowUpdatingEventHandler Delegate

The `OracleRowUpdatingEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdating` event.

Declaration

```
// C#
public delegate void OracleRowUpdatingEventHandler (object sender,
 OracleRowUpdatingEventArgs eventArgs);
```

Parameters

- *sender*

  The source of the event.

- *eventArgs*

  The `OracleRowUpdatingEventArgs` object that contains the event data.

Remarks

Event callbacks can be registered through this event delegate for applications that wish to be notified after a row is updated.

In the .NET framework, the convention of an event delegate requires two parameters: the object that raises the event and the event data.

See Also:

- Oracle.DataAccess.Client Namespace
- "OracleDataAdapter Class" on page 4-86

OracleTransaction Class

An `OracleTransaction` object represents a local transaction.

Class Inheritance

```
Object
  MarshalByRefObject
    OracleTransaction
```

Declaration

```
// C#
public sealed class OracleTransaction : MarshalByRefObject,
    IDbTransaction, IDisposable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

The application calls `BeginTransaction` on the `OracleConnection` object to create an `OracleTransaction` object. The `OracleTransaction` object can be created in one of the following two modes:

- Read Committed (default)
- Serializable

Any other mode results in an exception.

The execution of a **DDL** statement in the context of a transaction is not recommended since it results in an implicit commit that is not reflected in the state of the `OracleTransaction` object.

All operations related to **savepoint**s pertain to the current local transaction. Operations like commit and rollback performed on the transaction have no effect on data in any existing `DataSet`.

Example

```
// C#
//  Starts a transaction and inserts one record. If insert fails, rolls back
```

```
//  the transaction. Otherwise, commits the transaction.

...
string ConStr = "User Id=myschema;Password=mypassword;" +
     "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Create an OracleCommand object using the connection object
OracleCommand cmd = new OracleCommand("", con);

// Start a transaction
OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

try
  {
    cmd.CommandText = "insert into mytable values (99, 'foo')";
    cmd.CommandType = CommandType.Text;
    cmd.ExecuteNonQuery();
    txn.Commit();
    Console.WriteLine("One record is inserted into the database table.");
  }
    catch(Exception e)
  {
    txn.Rollback();
    Console.WriteLine("No record was inserted into the database table.");
  }
...
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Members
- OracleTransaction Static Methods
- OracleTransaction Properties

OracleTransaction Members

`OracleTransaction` members are listed in the following tables:

OracleTransaction Static Methods

`OracleTransaction` static methods are listed in Table 4–112.

Table 4–112 OracleTransaction Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleTransaction Properties

`OracleTransaction` properties are listed in Table 4–113.

Table 4–113 OracleTransaction Properties

Name	Description
IsolationLevel	Specifies the isolation level for the transaction
Connection	Specifies the connection that is associated with the transaction

OracleTransaction Public Methods

`OracleTransaction` public methods are listed in Table 4–114.

Table 4–114 OracleTransaction Public Methods

Public Method	Description
Commit	Commits the database transaction
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Frees the resources used by the `OracleTransaction` object
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`

Table 4–114 OracleTransaction Public Methods (Cont.)

Public Method	Description
Rollback	Rolls back a database transaction (Overloaded)
Save	Creates a savepoint within the current transaction
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class

OracleTransaction Static Methods

`OracleTransaction` static methods are listed in Table 4–115.

Table 4–115 OracleTransaction Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

OracleTransaction Properties

`OracleTransaction` properties are listed in Table 4–116.

Table 4–116 OracleTransaction Properties

Name	Description
IsolationLevel	Specifies the isolation level for the transaction
Connection	Specifies the connection that is associated with the transaction

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

IsolationLevel

This property specifies the isolation level for the transaction.

Declaration

```
// C#
public IsolationLevel IsolationLevel {get;}
```

Property Value

`IsolationLevel`

Implements

`IDbTransaction`

Exceptions

`InvalidOperationException` - The transaction has already completed.

Remarks

Default = `IsolationLevel.ReadCommitted`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

Connection

This property specifies the connection that is associated with the transaction.

Declaration

```
// C#
public OracleConnection Connection {get;}
```

Property Value

`Connection`

Implements

`IDbTransaction`

Remarks

This property indicates the `OracleConnection` object that is associated with the transaction.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

OracleTransaction Public Methods

`OracleTransaction` public methods are listed in Table 4–117.

Table 4–117 OracleTransaction Public Methods

Public Method	Description
Commit	Commits the database transaction
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Frees the resources used by the `OracleTransaction` object
`Equals`	Inherited from `Object` (Overloaded)
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
Rollback	Rolls back a database transaction (Overloaded)
Save	Creates a savepoint within the current transaction
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

Commit

This method commits the database transaction.

Declaration

```
// C#
public void Commit();
```

Implements

`IDbTransaction`

Exceptions

`InvalidOperationException` - The transaction has already been completed successfully, has been rolled back, or the associated connection is closed.

Remarks

Upon a successful commit, the transaction enters a completed state.

Example

```
// C#
//  Starts a transaction and inserts one record. If insert fails, rolls back
//  the transaction. Otherwise, commits the transaction.

...
string ConStr = "User Id=myschema;Password=mypassword;" +
     "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

//Create an OracleCommand object using the connection object
OracleCommand cmd = new OracleCommand("", con);

// Start a transaction
OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);
```

```
try
  {
    cmd.CommandText = "insert into mytable values (99, 'foo')";
    cmd.CommandType = CommandType.Text;
    cmd.ExecuteNonQuery();
    txn.Commit();
    Console.WriteLine("One record was inserted into the database table.");
  }
    catch(Exception e)
  {
    txn.Rollback();
    Console.WriteLine("No record was inserted into the database table.");
  }
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

Dispose

This method frees the resources used by the `OracleTransaction` object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

This method releases both the managed and unmanaged resources held by the `OracleTransaction` object. If the transaction is not in a completed state, an attempt to rollback the transaction is made.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

Rollback

`Rollback` rolls back a database transaction.

Overload List:

- Rollback()

  This method rolls back a database transaction.

- Rollback(string)

  This method rolls back a database transaction to a **savepoint** within the current transaction.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

Rollback()

This method rolls back a database transaction.

Declaration

```
// C#
public void Rollback();
```

Implements

`IDbTransaction`

Exceptions

`InvalidOperationException` - The transaction has already been completed successfully, has been rolled back, or the associated connection is closed.

Remarks

After a `Rollback()`, the `OracleTransaction` object can no longer be used because the `Rollback` ends the transaction.

Example

```
// C#
//  Starts a transaction and inserts one record. Then rolls back the
//  transaction.

...
string ConStr = "User Id=myschema;Password=mypassword;" +
   "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

OracleCommand cmd = Con.CreateCommand();

// Start a transaction
OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

cmd.CommandText = "insert into mytable values (99, 'foo')";
cmd.CommandType = CommandType.Text;
cmd.ExecuteNonQuery();
txn.Rollback();
Console.WriteLine("Nothing was inserted into the database table.");
...
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

Rollback(string)

This method rolls back a database transaction to a **savepoint** within the current transaction.

Declaration

```
// C#
public void Rollback(string savepointName);
```

Parameters

- *savepointName*

  The name of the savepoint to rollback to, in the current transaction.

Exceptions

`InvalidOperationException` - The transaction has already been completed successfully, has been rolled back, or the associated connection is closed.

Remarks

After a rollback to a savepoint, the current transaction remains active and can be used for further operations.

The *savepointName* specified does not have to match the case of the `savepointName` created using the `Save` method, since savepoints are created in the database in a case-insensitive manner.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

Save

This method creates a **savepoint** within the current transaction.

Declaration

```
// C#
public void Save(string savepointName);
```

Parameters

- *savepointName*

  The name of the savepoint being created in the current transaction.

Exceptions

`InvalidOperationException` - The transaction has already been completed.

Remarks

After creating a savepoint, the transaction does not enter a completed state and can be used for further operations.

The *savepointName* specified is created in the database in a case-insensitive manner. Calling the `Rollback` method rolls back to *savepointName*. This allows portions of a transaction to be rolled back, instead of the entire transaction.

Example

```
// C#
//  Starts a transaction and inserts two records. Creates a savepoint
//  within the current transaction for the first insert. Then rolls back to
//  the savepoint to commit the first record.

...
string ConStr = "User Id=myschema;Password=mypassword;" +
     "Data Source=oracle;";
OracleConnection con = new OracleConnection(ConStr);
con.Open();

OracleCommand cmd = Con.CreateCommand();

// Start a transaction
OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

cmd.CommandText = "insert into mytable values (99, 'foo')";
cmd.CommandType = CommandType.Text;
cmd.ExecuteNonQuery();

//Create a savepoint
txn.Save("MySavePoint");

cmd.CommandText = "insert into mytable values (100, 'bar')";
cmd.ExecuteNonQuery();

//Rollback to the savepoint
txn.Rollback("MySavePoint");

//Commit the first insert
txn.Commit();
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleTransaction Class
- OracleTransaction Members

OracleXmlQueryProperties Class

An `OracleXmlQueryProperties` object represents the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Query`.

Class Inheritance

```
Object

  OracleXmlQueryProperties
```

Declaration

```
public sealed class OracleXmlQueryProperties : ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

`OracleXmlQueryProperties` can be accessed, and modified using the `XmlQueryProperties` property of the `OracleCommand` class. Each `OracleCommand` object has its own instance of the `OracleXmlQueryProperties` class in the `XmlQueryProperties` property.

Use the default constructor to get a new instance of the `OracleXmlQueryProperties`. Use the `OracleXmlQueryProperties.Clone()` method to get a copy of an `OracleXmlQueryProperties` instance.

Example

This example retrieves relational data as XML.

```
// C#
 StreamReader sr = null;

// Create the connection.
string constr = "User Id=hr;Password=hr;Data Source=orcl";
OracleConnection conn = new OracleConnection(constr);
conn.Open();

// Create the command.
OracleCommand cmd = new OracleCommand("", conn);
```

```
// Set the XML command type to query.
cmd.XmlCommandType =  OracleXmlCommandType.Query;

// Set the SQL query.
cmd.CommandText = "select * from employees e where e.employee_id = :empno";

// Set command properties that affect XML query behaviour.
cmd.BindByName = true;
cmd.AddRowid = true;

// Bind values to the parameters in the SQL query.
Int32 empNum = 205;
cmd.Parameters.Add(":empno", OracleDbType.Int32, empNum,
        ParameterDirection.Input);

// Set the XML query properties.
cmd.XmlQueryProperties.MaxRows =  -1;
cmd.XmlQueryProperties.RootTag =  "MYROWSET";
cmd.XmlQueryProperties.RowTag =  "MYROW";
cmd.XmlQueryProperties.Xslt =  null;
cmd.XmlQueryProperties.XsltParams =  null;

// Test query execution without returning a result.
int rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Get the XML document as an XmlReader.
XmlReader xmlReader =  cmd.ExecuteXmlReader();
XmlDocument xmlDocument = new XmlDocument();
xmlDocument.PreserveWhitespace = true;
xmlDocument.Load(xmlReader);
Console.WriteLine(xmlDocument.OuterXml);

// Change the SQL query, and set the maximum number of rows to 2.
cmd.CommandText = "select * from employees e";
cmd.Parameters.Clear();
cmd.XmlQueryProperties.MaxRows =  2;

// Get the XML document as a Stream.
Stream stream = cmd.ExecuteStream();
sr = new StreamReader(stream, Encoding.Unicode);
Console.WriteLine(sr.ReadToEnd());

// Get all the rows.
```

```
cmd.XmlQueryProperties.MaxRows =  -1;

// Append the XML document to an existing Stream.
MemoryStream mstream = new MemoryStream(32);
cmd.ExecuteToStream(mstream);
mstream.Seek(0, SeekOrigin.Begin);
sr = new StreamReader(mstream, Encoding.Unicode);
Console.WriteLine(sr.ReadToEnd());

// Clean up.
cmd.Dispose();
conn.Close();
conn.Dispose();
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Members
- OracleXmlQueryProperties Constructor
- OracleXmlQueryProperties Properties
- OracleXmlQueryProperties Public Methods
- OracleCommand Class

OracleXmlQueryProperties Members

`OracleXmlQueryProperties` members are listed in the following tables:

OracleXmlQueryProperties Constructors

The `OracleXmlQueryProperties` constructors are listed in Table 4–118.

Table 4–118 OracleXmlQueryProperties Constructors

Constructor	Description
OracleXmlQueryProperties Constructor	Instantiates a new instance of the `OracleXmlQueryProperties` class

OracleXmlQueryProperties Properties

The `OracleXmlQueryProperties` properties are listed in Table 4–119.

Table 4–119 OracleXmlQueryProperties Properties

Name	Description
MaxRows	Specifies the maximum number of rows from the result set of the query that can be represented in the result XML document
RootTag	Specifies the root element of the result XML document
RowTag	Specifies the value of the XML element which identifies a row of data from the result set in an XML document
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies parameters for the XSL document

OracleXmlQueryProperties Public Methods

The `OracleXmlQueryProperties` public methods are listed in Table 4–120.

Table 4–120 OracleXmlQueryProperties Public Methods

Name	Description
Clone	Creates a copy of an `OracleXmlQueryProperties` object

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class

OracleXmlQueryProperties Constructor

The `OracleXmlQueryProperties` constructor instantiates a new instance of the `OracleXmlQueryProperties` class.

Declaration

```
// C#
public OracleXmlQueryProperties();
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
- OracleCommand Class

OracleXmlQueryProperties Properties

The `OracleXmlQueryProperties` properties are listed in Table 4–121.

Table 4–121 OracleXmlQueryProperties Properties

Name	Description
MaxRows	Specifies the maximum number of rows from the result set of the query that can be represented in the result XML document
RootTag	Specifies the root element of the result XML document
RowTag	Specifies the value of the XML element which identifies a row of data from the result set in an XML document
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies parameters for the XSL document

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
- OracleCommand Class

MaxRows

This property specifies the maximum number of rows from the result set of the query that can be represented in the result XML document.

Declaration

```
// C#
```

```
public int MaxRows {get; set;}
```

Property Value

The maximum number of rows.

Exceptions

`ArgumentException` - The new value for `MaxRows` is not valid.

Remarks

Default value is `-1`.

Possible values are:

- `-1` (all rows).
- A number greater than or equal to `0`.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
- OracleCommand Class

RootTag

This property specifies the root element of the result XML document.

Declaration

```
// C#
public string RootTag {get; set;}
```

Property Value

The root element of the result XML document.

Remarks

The default root tag is `ROWSET`.

To indicate that no root tag is be used in the result XML document, set this property to `null` or `""` or `String.Empty`.

If both `RootTag` and `RowTag` are set to `null`, an XML document is returned only if the result set returns one row and one column.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
- OracleCommand Class

RowTag

This property specifies the value of the XML element which identifies a row of data from the result set in an XML document.

Declaration

```
// C#
public string RowTag {get; set;}
```

Property Value

The value of the XML element.

Remarks

The default is `ROW`.

To indicate that no row tag is be used in the result XML document, set this property to `null` or `""` or `String.Empty`.

If both `RootTag` and `RowTag` are set to `null`, an XML document is returned only if the result set returns one row and one column.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
- OracleCommand Class

XsIt

This property specifies the XSL document used for XML transformation using XSLT.

Declaration

```
// C#
public string Xslt {get; set;}
```

Property Value

The XSL document used for XML transformation.

Remarks

Default value is `null`.

The XSL document is used for XML transformation of the XML document generated from the result set of the query.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
- OracleCommand Class

XsltParams

This property specifies parameters for the XSL document.

Declaration

```
// C#
public string XsltParams {get; set;}
```

Property Value

The parameters for the XSL document.

Remarks

Default value is `null`.

The parameters are specified as a string of "`name=value`" pairs of the form "`param1=value1; param2=value2; ...`" delimited by semicolons.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
- OracleCommand Class

OracleXmlQueryProperties Public Methods

The `OracleXmlQueryProperties` public methods are listed in Table 4–122.

Table 4–122 OracleXmlQueryProperties Public Methods

Name	Description
Clone	Creates a copy of an `OracleXmlQueryProperties` object

Clone

This method creates a copy of an `OracleXmlQueryProperties` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleXmlQueryProperties` object

Implements

`ICloneable`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
- OracleCommand Class

OracleXmlSaveProperties Class

An `OracleXmlSaveProperties` object represents the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Insert`, `Update`, or `Delete`.

Class Inheritance

```
Object

  OracleXmlSaveProperties
```

Declaration

```
public sealed class OracleXmlSaveProperties : ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

`OracleXmlSaveProperties` can be accessed and modified using the `XmlSaveProperties` property of the `OracleCommand` class. Each `OracleCommand` object has its own instance of the `OracleXmlSaveProperties` class in the `XmlSaveProperties` property.

Use the default constructor to get a new instance of `OracleXmlSaveProperties`. Use the `OracleXmlSaveProperties.Clone()` method to get a copy of an `OracleXmlSaveProperties` instance.

Example

This sample demonstrates how to do inserts, updates, and deletes to a relational table or view using an XML document.

```
// C#
string[] KeyColumnsList = null;
string[] UpdateColumnsList = null;
int rows = 0;

// Create the connection.
string constr = "User Id=hr;Password=hr;Data Source=orcl";
```

```
OracleConnection conn = new OracleConnection(constr);
conn.Open();

// Create the command.
OracleCommand cmd = new OracleCommand("", conn);

// Set the XML command type to insert.
cmd.XmlCommandType = OracleXmlCommandType.Insert;

// Set the XML document.
cmd.CommandText = "<?xml version=\"1.0\"?>\n" +
                  "<ROWSET>\n" +
                  "  <MYROW num = \"1\">\n" +
                  "    <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
                  "    <LAST_NAME>Smith</LAST_NAME>\n" +
                  "    <EMAIL>Smith@Oracle.com</EMAIL>\n" +
                  "    <HIRE_DATE>1/1/2003 0:0:0</HIRE_DATE>\n" +
                  "    <JOB_ID>IT_PROG</JOB_ID>\n" +
                  "  </MYROW>\n" +
                  "  <MYROW num = \"2\">\n" +
                  "    <EMPLOYEE_ID>1235</EMPLOYEE_ID>\n" +
                  "    <LAST_NAME>Barney</LAST_NAME>\n" +
                  "    <EMAIL>Barney@Oracle.com</EMAIL>\n" +
                  "    <HIRE_DATE>1/1/2003 0:0:0</HIRE_DATE>\n" +
                  "    <JOB_ID>IT_PROG</JOB_ID>\n" +
                  "  </MYROW>\n" +
                  "</ROWSET>\n";

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";

UpdateColumnsList = new string[5];
UpdateColumnsList[0] = "EMPLOYEE_ID";
UpdateColumnsList[1] = "LAST_NAME";
UpdateColumnsList[2] = "EMAIL";
UpdateColumnsList[3] = "HIRE_DATE";
UpdateColumnsList[4] = "JOB_ID";

cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
cmd.XmlSaveProperties.RowTag =  "MYROW";
cmd.XmlSaveProperties.Table =  "employees";
cmd.XmlSaveProperties.UpdateColumnsList = UpdateColumnsList;
cmd.XmlSaveProperties.Xslt =  null;
cmd.XmlSaveProperties.XsltParams =  null;
```

```
// Do the inserts.
rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Set the XML command type to update.
cmd.XmlCommandType =  OracleXmlCommandType.Update;

// Set the XML document.
cmd.CommandText = "<?xml version=\"1.0\"?>\n" +
                  "<ROWSET>\n" +
                  "  <MYROW num = \"1\">\n" +
                  "    <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
                  "    <LAST_NAME>Adams</LAST_NAME>\n" +
                  "  </MYROW>\n" +
                  "</ROWSET>\n";

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";

UpdateColumnsList = new string[1];
UpdateColumnsList[0] = "LAST_NAME";

cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
cmd.XmlSaveProperties.UpdateColumnsList = UpdateColumnsList;

// Do the updates.
rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Set the XML command type to delete.
cmd.XmlCommandType =  OracleXmlCommandType.Delete;

// Set the XML document.
cmd.CommandText = "<?xml version=\"1.0\"?>\n" +
                  "<ROWSET>\n" +
                  "  <MYROW num = \"1\">\n" +
                  "    <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
                  "  </MYROW>\n" +
                  "  <MYROW num = \"2\">\n" +
                  "    <EMPLOYEE_ID>1235</EMPLOYEE_ID>\n" +
                  "  </MYROW>\n" +
                  "</ROWSET>\n";
```

```
// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";

cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
cmd.XmlSaveProperties.UpdateColumnsList = null;

// Do the deletes.
rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Clean up.
cmd.Dispose();
conn.Close();
conn.Dispose();
```

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Members
- OracleXmlSaveProperties Constructor
- OracleXmlSaveProperties Properties
- OracleXmlSaveProperties Public Methods
- OracleCommand Class

OracleXmlSaveProperties Members

`OracleXmlSaveProperties` members are listed in the following tables:

OracleXmlSaveProperties Constructor

`OracleXmlSaveProperties` constructors are listed in Table 4–123

Table 4–123 OracleXmlSaveProperties Constructor

Constructor	Description
OracleXmlSaveProperties Constructor	Instantiates a new instance of the `OracleXmlSaveProperties` class

OracleXmlSaveProperties Properties

The `OracleXmlSaveProperties` properties are listed in Table 4–124.

Table 4–124 OracleXmlSaveProperties Properties

Name	Description
KeyColumnsList	Specifies the list of columns used as a key to locate existing rows for update or delete using an XML document
RowTag	Specifies the value for the XML element that identifies a row of data in an XML document
Table	Specifies the name of the table or view to which changes are saved
UpdateColumnsList	Specifies the list of columns to update or insert
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies the parameters for the XSLT document specified in the Xslt property

OracleXmlSaveProperties Public Methods

The `OracleXmlSaveProperties` public methods are listed in Table 4–125.

Table 4–125 OracleXmlSaveProperties Public Methods

Name	Description
Clone	Creates a copy of an `OracleXmlSaveProperties` object

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

OracleXmlSaveProperties Constructor

The `OracleXmlSaveProperties` constructor instantiates a new instance of `OracleXmlSaveProperties` class.

Declaration

```
// C#
public OracleXmlSaveProperties;
```

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

OracleXmlSaveProperties Properties

The `OracleXmlSaveProperties` properties are listed in Table 4–126.

Table 4–126 OracleXmlSaveProperties Properties

Name	Description
KeyColumnsList	Specifies the list of columns used as a key to locate existing rows for update or delete using an XML document
RowTag	Specifies the value for the XML element that identifies a row of data in an XML document
Table	Specifies the name of the table or view to which changes are saved
UpdateColumnsList	Specifies the list of columns to update or insert
Xslt	Specifies the XSL document used for XML transformation using XSLT
XsltParams	Specifies the parameters for the XSLT document specified in the Xslt property

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

KeyColumnsList

This property specifies the list of columns used as a key to locate existing rows for update or delete using an XML document.

Declaration

```
// C#
public string[] KeyColumnsList {get; set;}
```

Property Value

The list of columns.

Remarks

Default value is null.

The first null value (if any) terminates the list.

`KeyColumnsList` usage with `XMLCommandType` property values:

- `Insert` - `KeyColumnsList` is ignored and can be null.
- `Update` - `KeyColumnsList` must be specified; it identifies the columns to use to find the rows to be updated.
- `Delete` - If `KeyColumnsList` is null, all the column values in each row element in the XML document are used to locate the rows to delete. Otherwise, `KeyColumnsList` specifies the columns used to identify the rows to delete.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

RowTag

This property specifies the value for the XML element that identifies a row of data in an XML document.

Declaration

```
// C#
public string RowTag {get; set;}
```

Property Value

An XML element name.

Remarks

The default value is `ROW`.

Each element in the XML document identifies one row in a table or view.

If `RowTag` is set to `""` or `null`, no row tag is used in the XML document. In this case, the XML document is assumed to contain only one row.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

Table

This property specifies the name of the table or view to which changes are saved.

Declaration

```
// C#
public string Table {get; set;}
```

Property Value

A table name.

Remarks

Default value is `null`.

The property must be set to a valid table or view name.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

UpdateColumnsList

This property specifies the list of columns to update or insert.

Declaration

```
// C#
public string[] UpdateColumnsList {get; set;}
```

Property Value

A list of columns.

Remarks

Default value is null.

The first null value (if any) terminates the list.

`UpdateColumnList` usage with `XMLCommandType` property values:

- `Insert` - `UpdateColumnList` indicates which columns are assigned values when a new row is created. If `UpdateColumnList` is null, then all columns are assigned values. If a column is on the `UpdateColumnList`, but no value is specified for the row in the XML file, then `NULL` is used. If a column is not on the `UpdateColumnList`, then the default value for that column is used.
- `Update` - `UpdateColumnList` specifies columns to modify for each row of data in the XML document. If `UpdateColumnList` is null, all the values in each XML element in the XML document are used to modify the columns.
- `Delete` - `UpdateColumnsList` is ignored and can be null.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

XsIt

This property specifies the XSL document used for XML transformation using XSLT.

Declaration

```
// C#
public string Xslt {get; set;}
```

Property Value

The XSL document used for XML transformation.

Remarks

Default = `null`.

The XSL document is used for XSLT transformation of a given XML document. The transformed XML document is used to save changes to the table or view.

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

XsltParams

This property specifies the parameters for the XSLT document specified in the `Xslt` property.

Declaration

```
// C#
public string XsltParams {get; set;}
```

Property Value

The parameters for the XSLT document .

Remarks

Default is `null`.

This property is a string delimited by semicolons in "`name=value`" pairs of the form "`param1=value1; param2=value2; ...`".

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

OracleXmlSaveProperties Public Methods

The `OracleXmlSaveProperties` public methods are listed in Table 4–127.

Table 4–127 OracleXmlSaveProperties Public Methods

Name	Description
Clone	Creates a copy of an `OracleXmlSaveProperties` object

Clone

This method creates a copy of an `OracleXmlSaveProperties` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleXmlSaveProperties` object

Implements

`ICloneable`

See Also:

- Oracle.DataAccess.Client Namespace
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

Oracle Data Provider Enumerations

The following is a list of enumerations that Oracle Data Provider for .NET provides.

- FailoverEvent Enumeration
- FailoverReturnCode Enumeration
- FailoverType Enumeration
- OracleCollectionType Enumeration
- OracleDbType Enumeration
- OracleParameterStatus Enumeration
- OracleXmlCommandType Enumeration

FailoverEvent Enumeration

`FailoverEvent` enumerated values are used to explicitly specify the state of the failover.

Table 4–128 lists all the `FailoverEvent` enumeration values with a description of each enumerated value.

Table 4–128 FailoverEvent Enumeration Values

Member Names	Description
`FailoverEvent.Begin`	Indicates that failover has detected a lost connection and that failover is starting.
`FailoverEvent.End`	Indicates successful completion of failover.
`FailoverEvent.Abort`	Indicates that failover was unsuccessful, and there is no option of retrying.
`FailoverEvent.Error`	Indicates that failover was unsuccessful, and it gives the application the opportunity to handle the error and retry failover. The application can retry failover by returning `FailoverReturnCode.Retry` for the event notification.
`FailoverEvent.Reauth`	Indicates that a user handle has been reauthenticated. This applies to the situation where a client has multiple user sessions on a single server connection. During the initial failover, only the active user session is failed over. Other sessions are failed over when the application tries to use them. This is the value passed to the callback during these subsequent failovers.

See Also:

- Oracle Data Provider Enumerations
- "OracleFailoverEventArgs Class" on page 4-204
- "FailoverEvent" on page 4-208
- *Oracle Real Application Clusters Quick Start*
- *Oracle Net Services Reference Guide*

FailoverReturnCode Enumeration

`FailoverReturnCode` enumerated values are passed back by the application to the ODP.NET provider to request a retry in case of a failover error or to continue in case of a successful failover.

Table 4–129 lists the `FailoverReturnCode` enumeration values with a description of each enumerated value.

Table 4–129 FailoverReturnCode Enumeration Values

Member Names	Description
`FailoverReturnCode.Retry`	Requests ODP.NET to retry failover in case `FailoverEvent.Error` is passed to the application
`FailoverReturnCode.Success`	Requests ODP.NET to proceed so that the application receive more notifications, if any

See Also:

- Oracle Data Provider Enumerations
- "OracleFailoverEventArgs Class" on page 4-204
- "FailoverEvent" on page 4-208
- *Oracle Real Application Clusters Quick Start*
- *Oracle Net Services Reference Guide*

FailoverType Enumeration

`FailoverType` enumerated values are used to indicate the type of failover event that was raised.

Table 4–130 lists all the `FailoverType` enumeration values with a description of each enumerated value.

Table 4–130 FailoverType Enumeration Values

Member Names	Description
`FailoverType.Session`	Indicates that the user has requested only session failover.
`FailoverType.Select`	Indicates that the user has requested select and session failover.

See Also:

- Oracle Data Provider Enumerations
- "OracleFailoverEventArgs Class" on page 4-204
- "FailoverType" on page 4-207
- *Oracle Real Application Clusters Quick Start*
- *Oracle Net Services Reference Guide*

OracleCollectionType Enumeration

`OracleCollectionType` enumerated values specify whether the `OracleParameter` object represents a collection, and if so, specifies the collection type.

Table 4–131 lists all the `OracleCollectionType` enumeration values with a description of each enumerated value.

Table 4–131 OracleCollectionType Enumeration Values

Member Name	Description
`None`	Is not a collection type
`PLSQLAssociativeArray`	Indicates that the collection type is a PL/SQL Associative Array (or PL/SQL Index-By Table)

See Also:

- Oracle Data Provider Enumerations
- "OracleParameter Class" on page 4-244
- "CollectionType" on page 4-265

OracleDbType Enumeration

`OracleDbType` enumerated values are used to explicitly specify the `OracleDbType` of an `OracleParameter`.

Table 4–132 lists all the `OracleDbType` enumeration values with a description of each enumerated value.

Table 4–132 OracleDbType Enumeration Values

Member Name	Description
`BFile`	Oracle `BFILE` type
`Blob`	Oracle `BLOB` type
`Byte`	`byte` type
`Char`	Oracle `CHAR` type
`Clob`	Oracle `CLOB` type
`Date`	Oracle `DATE` type
`Decimal`	Oracle `NUMBER` type
`Double`	8-byte `FLOAT` type
`Int16`	2-byte `INTEGER` type
`Int32`	4-byte `INTEGER` type
`Int64`	8-byte `INTEGER` type
`IntervalDS`	Oracle `INTERVAL DAY TO SECOND` type
`IntervalYM`	Oracle `INTERVAL YEAR TO MONTH` type
`Long`	Oracle `LONG` type
`LongRaw`	Oracle `LONG RAW` type
`NChar`	Oracle `NCHAR` type
`NClob`	Oracle `NCLOB` type
`NVarchar2`	Oracle `NVARCHAR2` type
`Raw`	Oracle `RAW` type
`RefCursor`	Oracle `REF CURSOR` type
`Single`	4-byte `FLOAT` type

Table 4–132 OracleDbType Enumeration Values (Cont.)

Member Name	Description
`TimeStamp`	Oracle `TIMESTAMP` type
`TimeStampLTZ`	Oracle `TIMESTAMP WITH LOCAL TIME ZONE` type
`TimeStampTZ`	Oracle `TIMESTAMP WITH TIME ZONE` type
`Varchar2`	Oracle `VARCHAR2` type
`XmlType`	Oracle `XMLType` type

See Also:

- Oracle Data Provider Enumerations
- "OracleParameter Class" on page 4-244
- "OracleParameterCollection Class" on page 4-281
- `OracleParameter` "OracleDbType" on page 4-268

OracleParameterStatus Enumeration

The `OracleParameterStatus` enumeration type indicates whether a `NULL` value is fetched from a column, whether truncation has occurred during the fetch, or whether a `NULL` value is to be inserted into a database column.

Table 4–133 lists all the `OracleParameterStatus` enumeration values with a description of each enumerated value.

Table 4–133 OracleParameterStatus Members

Member Name	Description
`Success`	Indicates that (for input parameters) the input value has been assigned to the column. For output parameter, it indicates that the provider assigned an intact value to the parameter.
`NullFetched`	Indicates that a `NULL` value has been fetched from a column or an `OUT` parameter
`NullInsert`	Indicates that a `NULL` value is to be inserted into a column
`Truncation`	Indicates that truncation has occurred when fetching the data from the column

See Also:

- Oracle Data Provider Enumerations
- "OracleParameter Class" on page 4-244
- `OracleParameter` "ArrayBindStatus" on page 4-264
- `OracleParameter` "Value" on page 4-276

OracleXmlCommandType Enumeration

The `OracleXmlCommandType` enumeration specifies the values that are allowed for the `XmlCommandType` property of the `OracleCommand` class. It is used to specify the type of XML operation.

Table 4–134 lists all the `OracleXmlCommandType` enumeration values with a description of each enumerated value.

Table 4–134 OracleXmlCommandType Members

Member Name	Description
`None`	No XML operation is desired
`Query`	The command text is a SQL query and the result of the query is an XML document. The SQL query needs to be a select statement
`Insert`	The command text is an XML document containing rows to insert.
`Update`	The command text is an XML document containing rows to update.
`Delete`	The command text is an XML document containing rows to delete.

See Also:

- Oracle Data Provider Enumerations
- OracleCommand Class
- "XmlCommandType" on page 4-25

5

Oracle.DataAccess.Types Namespace (ODP.NET Types)

This chapter describes the type structures and objects provided by Oracle Data Provider for .NET.

This chapter contains these topics:

Overview of ODP.NET Types

This chapter describes the Oracle Data Provider for .NET Types (ODP.NET Types). The ODP.NET Types are located in the `Oracle.DataAccess.Types` Namespace and composed of the following:

- ODP.NET Type Structures
- ODP.NET Type Objects
  - ODP.NET LOB Types (`OracleBFile`, `OracleBlob`, `OracleClob` objects)
  - `OracleRefCursor` Objects
  - `OracleXmlStream` Objects
  - `OracleXmlType` Objects
- ODP.NET Type Exceptions

Syntax Used

The class descriptions use the C# syntax and datatypes. Check the related Visual Studio .NET Framework documentation for information on Visual Basic syntax.

Assembly and Namespace

ODP.NET Types are provided in the `Oracle.DataAccess.Types` namespace of the `Oracle.DataAccess.dll` assembly.

The client interfaces are located in the following sections:

- ODP.NET Type Structures
- ODP.NET Type Exceptions
- ODP.NET Type Objects

ODP.NET Type Structures

This section covers the ODP.NET Type structures.

OracleBinary Structure

The `OracleBinary` structure represents a variable-length stream of binary data to be stored in or retrieved from a database.

Class Inheritance

```
Object
  ValueType
    OracleBinary
```

Declaration

```
// C#
public struct OracleBinary : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
// Concatenation of the OracleBinary values using + operator
OracleBinary ob1a = new OracleBinary(new byte[] {1,2,3});
OracleBinary ob1b = new OracleBinary(new byte[] {4});
OracleBinary ob1 = ob1a + ob1b;
OracleBinary ob2 = new OracleBinary(new byte[] {1,2,3,4});
// Output the length if ob1 is equal to ob2
if (ob1 == ob2)
    Console.WriteLine("The OracleBinary structures are equal with length "
         + ob1.Length);
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Members
- OracleBinary Constructor
- OracleBinary Static Fields
- OracleBinary Static Methods
- OracleBinary Static Operators
- OracleBinary Static Type Conversion Operators
- OracleBinary Properties
- OracleBinary Instance Methods

OracleBinary Members

`OracleBinary` members are listed in the following tables:

OracleBinary Constructors

`OracleBinary` constructors are listed in Table 5–1

Table 5–1 OracleBinary Constructors

Constructor	Description
OracleBinary Constructor	Instantiates a new instance of `OracleBinary` structure

OracleBinary Static Fields

The `OracleBinary` static fields are listed in Table 5–2.

Table 5–2 OracleBinary Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the `OracleBinary` structure

OracleBinary Static Methods

The `OracleBinary` static methods are listed in Table 5–3.

Table 5–3 OracleBinary Static Methods

Methods	Description
Concat	Returns the concatenation of two `OracleBinary` structures
Equals	Determines if two `OracleBinary` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleBinary` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleBinary` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleBinary` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleBinary` values is less than or equal to the second
NotEquals	Determines if two `OracleBinary` values are not equal

OracleBinary Static Operators

The `OracleBinary` static operators are listed in Table 5–4.

Table 5–4 OracleBinary Static Operators

Operator	Description
operator +	Concatenates two `OracleBinary` values
operator ==	Determines if two `OracleBinary` values are equal
operator >	Determines if the first of two `OracleBinary` values is greater than the second
operator >=	Determines if the first of two `OracleBinary` values is greater than or equal to the second
operator !=	Determines if two `OracleBinary` values are not equal
operator <	Determines if the first of two `OracleBinary` value is less than the second
operator <=	Determines if the first of two `OracleBinary` value is less than or equal to the second

OracleBinary Static Type Conversion Operators

The `OracleBinary` static type conversion operators are listed in Table 5–5.

Table 5–5 OracleBinary Static Type Conversion Operators

Operator	Description
explicit operator byte[ ]	Converts an instance value to a byte array
implicit operator OracleBinary	Converts an instance value to an `OracleBinary` structure

OracleBinary Properties

The `OracleBinary` properties are listed in Table 5–6.

Table 5–6 OracleBinary Properties

Properties	Description
IsNull	Indicates whether the current instance has a null value
Item	Obtains the particular `byte` in an `OracleBinary` structure using an index
Length	Returns the length of the binary data
Value	Returns the binary data that is stored in an `OracleBinary` structure

OracleBinary Instance Methods

The `OracleBinary` instance methods are listed in Table 5–7.

Table 5–7 OracleBinary Instance Methods

Methods	Description
CompareTo	Compares the current instance to an object and returns an integer that represents their relative values
Equals	Determines if two objects contain the same binary data (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from `Object`
ToString	Converts the current `OracleBinary` structure to a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure

OracleBinary Constructor

The `OracleBinary` constructor instantiates a new instance of the `OracleBinary` structure and sets its value to the provided array of bytes.

Declaration

```
// C#
public OracleBinary(byte[ ] bytes);
```

Parameters

- *bytes*

  A byte array.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

OracleBinary Static Fields

The `OracleBinary` static fields are listed in Table 5–8.

Table 5–8 OracleBinary Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the `OracleBinary` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

Null

This static field represents a null value that can be assigned to an instance of the `OracleBinary` structure.

Declaration

```
// C#
public static readonly OracleBinary Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

OracleBinary Static Methods

The `OracleBinary` static methods are listed in Table 5–9.

Table 5–9 OracleBinary Static Methods

Methods	Description
Concat	Returns the concatenation of two `OracleBinary` structures
Equals	Determines if two `OracleBinary` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleBinary` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleBinary` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleBinary` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleBinary` values is less than or equal to the second
NotEquals	Determines if two `OracleBinary` values are not equal

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

Concat

This method returns the concatenation of two `OracleBinary` structures.

Declaration

```
// C#
public static OracleBinary Concat(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

An `OracleBinary`.

Remarks

If either argument has a null value, the returned `OracleBinary` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

Equals

This method determines if two `OracleBinary` values are equal.

Declaration

```
// C#
public static bool Equals(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if two `OracleBinary` values are equal; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

GreaterThan

This method determines whether the first of two `OracleBinary` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is greater than the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

Example

```
// C#
OracleBinary ob1 = OracleBinary.Null;
OracleBinary ob2 = new OracleBinary(new byte[] {1});
if (OracleBinary.GreaterThan(ob2,ob1))
  Console.WriteLine("ob2 > ob1");
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

GreaterThanOrEqual

This method determines whether the first of two `OracleBinary` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is greater than or equal to the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

LessThan

This method determines whether the first of two `OracleBinary` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is less than the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

LessThanOrEqual

This method determines whether the first of two `OracleBinary` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is less than or equal to the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

NotEquals

This method determines whether two `OracleBinary` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if two `OracleBinary` values are not equal; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

OracleBinary Static Operators

The `OracleBinary` static operators are listed in Table 5–10.

Table 5–10 OracleBinary Static Operators

Operator	Description
operator +	Concatenates two `OracleBinary` values

Table 5–10 OracleBinary Static Operators (Cont.)

Operator	Description
operator ==	Determines if two `OracleBinary` values are equal
operator >	Determines if the first of two `OracleBinary` values is greater than the second
operator >=	Determines if the first of two `OracleBinary` values is greater than or equal to the second
operator !=	Determines if two `OracleBinary` values are not equal
operator <	Determines if the first of two `OracleBinary` value is less than the second
operator <=	Determines if the first of two `OracleBinary` value is less than or equal to the second

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

operator +

This method concatenates two `OracleBinary` values.

Declaration

```
// C#
public static OracleBinary operator + (OracleBinary value1, OracleBinary
value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

`OracleBinary`

Remarks

If either argument has a null value, the returned `OacleBinary` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

operator ==

This method determines if two `OracleBinary` values are equal.

Declaration

```
// C#
public static bool operator == (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if they are the same; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinarys` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

operator >

This method determines if the first of two `OracleBinary` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

Example

```
// C#
OracleBinary ob1 = OracleBinary.Null;
OracleBinary ob2 = new OracleBinary(new byte[] {1});
if (ob2 > ob1)
  Console.WriteLine("ob2 > ob1");
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

operator >=

This method determines if the first of two `OracleBinary` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

operator !=

This method determines if two `OracleBinary` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the two `OracleBinary` values are not equal; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

operator <

This method determines if the first of two `OracleBinary` values is less than the second.

Declaration

```
// C#
public static bool operator < ( OracleBinary value1, OracleBinary value2);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

operator <=

This method determines if the first of two `OracleBinary` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleBinary value1, OracleBinary value1);
```

Parameters

- *value1*

  First `OracleBinary`.

- *value2*

  Second `OracleBinary`.

Return Value

Returns `true` if the first of two `OracleBinary` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

OracleBinary Static Type Conversion Operators

The `OracleBinary` static type conversion operators are listed in Table 5–11.

Table 5–11 OracleBinary Static Type Conversion Operators

Operator	Description
explicit operator byte[ ]	Converts an instance value to a byte array
implicit operator OracleBinary	Converts an instance value to an `OracleBinary` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

explicit operator byte[ ]

This method converts an `OracleBinary` value to a byte array.

Declaration

```
// C#
public static explicit operator byte[ ] (OracleBinary val);
```

Parameters

- *val*

  An `OracleBinary`.

Return Value

A byte array.

Exceptions

`OracleNullValueException` - The `OracleBinary` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

implicit operator OracleBinary

This method converts a byte array to an `OracleBinary` structure.

Declaration

```
// C#
public static implicit operator OracleBinary(byte[ ] bytes);
```

Parameters

- *bytes*

  A byte array.

Return Value

`OracleBinary`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

OracleBinary Properties

The `OracleBinary` properties are listed in Table 5–12.

Table 5–12 OracleBinary Properties

Properties	Description
IsNull	Indicates whether the current instance has a null value
Item	Obtains the particular `byte` in an `OracleBinary` structure using an index
Length	Returns the length of the binary data
Value	Returns the binary data that is stored in an `OracleBinary` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

IsNull

This property indicates whether the current instance has a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

Returns `true` if the current instance has a null value; otherwise returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

Item

This property obtains the particular `byte` in an `OracleBinary` structure using an index.

Declaration

```
// C#
public byte this[int index] {get;}
```

Property Value

A byte in the specified index.

Exceptions

`OracleNullValueException` - The current instance has a null value.

Example

```
// C#
OracleBinary ob1 = new OracleBinary(new byte[] {4,3,2,1});
Console.WriteLine(ob1[ob1.Length-1]); // Prints the value 1
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

Length

This property returns the length of the binary data.

Declaration

```
// C#
public int length {get;}
```

Property Value

Length of the binary data.

Exceptions

`OracleNullValueException` - The current instance has a null value.

Example

```
// C#
OracleBinary ob1 = new OracleBinary(new byte[] {4,3,2,1});
Console.WriteLine(ob1.Length); // Prints the value 4
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

Value

This property returns the binary data that is stored in the `OracleBinary` structure.

Declaration

```
// C#
public byte[] Value {get;}
```

Property Value

Binary data.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

OracleBinary Instance Methods

The `OracleBinary` instance methods are listed in Table 5–13.

Table 5–13 OracleBinary Instance Methods

Methods	Description
CompareTo	Compares the current instance to an object and returns an integer that represents their relative values
Equals	Determines if two objects contain the same binary data (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from `Object`

Table 5–13 OracleBinary Instance Methods (Cont.)

Methods	Description
ToString	Converts the current `OracleBinary` structure to a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

CompareTo

This method compares the current instance to an object and returns an integer that represents their relative values

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  The object being compared.

Return Value

The method returns a number that is:

- Less than zero: if the current `OracleBinary` instance value is less than *obj*.
- Zero: if the current `OracleBinary` instance and *obj* values have the same binary data.
- Greater than zero: if the current `OracleBinary` instance value is greater than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The parameter is not of type `OracleBinary`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleBinary`s. For example, comparing an `OracleBinary` instance with an `OracleTimeStamp` instance is not allowed. When an `OracleBinary` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

Example

```
// C#
OracleBinary ob1 = new OracleBinary(new byte[] {1,1,1,1});
OracleBinary ob2 = new OracleBinary(new byte[] {1});

// append to ob2 while they are not equal
while (ob1.CompareTo(ob2) != 0)
   ob2 = ob2 + ob2;
Console.WriteLine("ob1 == ob2");
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

Equals

This method determines whether an object is an instance of `OracleBinary`, and has the same binary data as the current instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *obj*

  The object being compared.

Return Value

Returns `true` if *`obj`* is an instance of `OracleBinary`, and has the same binary data as the current instance; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleBinary` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

An `int` that represents the hash.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

ToString

Overrides `Object`

This method converts an `OracleBinary` instance to a string instance.

Declaration

```
// C#
public override string ToString();
```

Return Value

`string`

Remarks

If the current `OracleBinary` instance has a null value, the returned string "null".

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBinary Structure
- OracleBinary Members

OracleDate Structure

The `OracleDate` structure represents the Oracle `DATE` datatype to be stored in or retrieved from a database. Each `OracleDate` stores the following information: year, month, day, hour, minute, and second.

Class Inheritance

```
Object
  ValueType
    OracleDate
```

Declaration

```
// C#
public struct OracleDate : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
// Compute the number of days between minimum values of
// the OracleDate and DateTime structures
OracleDate od1 = new OracleDate(DateTime.MinValue);
OracleDate od2 = OracleDate.MinValue;

// Set the nls_date_format for the ToString()
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.DateFormat = "DD-MON-YYYY BC";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine("DateTime.MinValue   = " + od1.ToString());
Console.WriteLine("OracleDate.MinValue = " + od2.ToString());

// Compare the two values
int result = od1.CompareTo(od2);

// Output the difference in number of days (as a positive value)
if (result == 0)
  Console.WriteLine("DateTime and OracleDate Minimum values"+
```

```
    " are equal.");
else if (result < 0)
  Console.WriteLine("DateTime Minimum value is before OracleDate" +
    " Minumum value by " + od2.GetDaysBetween(od1) + " days.");
else if (result > 0)
  Console.WriteLine("OracleDate Minimum value is before DateTime" +
    " Minumum value by " + od1.GetDaysBetween(od2) + " days.");
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Members
- OracleDate Constructors
- OracleDate Static Fields
- OracleDate Static Methods
- OracleDate Static Operators
- OracleDate Static Type Conversions
- OracleDate Properties
- OracleDate Methods

OracleDate Members

`OracleDate` members are listed in the following tables:

OracleDate Constructors

`OracleDate` constructors are listed in Table 5–14

Table 5–14 OracleDate Constructors

Constructor	Description
OracleDate Constructors	Instantiates a new instance of `OracleDate` structure (Overloaded)

OracleDate Static Fields

The `OracleDate` static fields are listed in Table 5–15.

Table 5–15 OracleDate Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleDate` structure, which is December 31, 9999 23:59:59
MinValue	Represents the minimum valid date for an `OracleDate` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to the value of an `OracleDate` structure instance

OracleDate Static Methods

The `OracleDate` static methods are listed in Table 5–16.

Table 5–16 OracleDate Static Methods

Methods	Description
Equals	Determines if two `OracleDate` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleDate` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleDate` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleDate` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleDate` values is less than or equal to the second
NotEquals	Determines if two `OracleDate` values are not equal
GetSysDate	Returns an `OracleDate` structure that represents the current date and time
Parse	Returns an `OracleDate` structure and sets its value using a string

OracleDate Static Operators

The `OracleDate` static operators are listed in Table 5–17.

Table 5–17 OracleDate Static Operators

Operator	Description
operator ==	Determines if two `OracleDate` values are the same
operator >	Determines if the first of two `OracleDate` values is greater than the second
operator >=	Determines if the first of two `OracleDate` values is greater than or equal to the second
operator !=	Determines if the two `OracleDate` values are not equal
operator <	Determines if the first of two `OracleDate` values is less than the second
operator <=	Determines if the first of two `OracleDate` values is less than or equal to the second

OracleDate Static Type Conversions

The `OracleDate` static type conversions are listed in Table 5–18.

Table 5–18 OracleDate Static Type Conversions

Operator	Description
explicit operator DateTime	Converts a structure to a `DateTime` structure
explicit operator OracleDate	Converts a structure to an `OracleDate` structure (Overloaded)

OracleDate Properties

The `OracleDate` properties are listed in Table 5–19.

Table 5–19 OracleDate Properties

Properties	Description
BinData	Gets an array of bytes that represents an Oracle `DATE` in Oracle internal format
Day	Gets the day component of an `OracleDate` method
IsNull	Indicates whether the current instance has a null value
Hour	Gets the `hour` component of an `OracleDate`
Minute	Gets the minute component of an `OracleDate`

Table 5–19 OracleDate Properties (Cont.)

Properties	Description
Month	Gets the `month` component of an `OracleDate`
Second	Gets the `second` component of an `OracleDate`
Value	Gets the date and time that is stored in the `OracleDate` structure
Year	Gets the `year` component of an `OracleDate`

OracleDate Methods

The `OracleDate` methods are listed in Table 5–20.

Table 5–20 OracleDate Methods

Methods	Description
CompareTo	Compares the current `OracleDate` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether an object has the same date and time as the current `OracleDate` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleDate` instance
GetDaysBetween	Calculates the number of days between the current `OracleDate` instance and an `OracleDate` structure
GetType	Inherited from `Object`
ToOracleTimeStamp	Converts the current `OracleDate` structure to an `OracleTimeStamp` structure
ToString	Converts the current `OracleDate` structure to a `string`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure

OracleDate Constructors

The `OracleDate` constructors instantiates a new instance of the `OracleDate` structure.

Overload List:

- OracleDate(DateTime)

  This constructor creates a new instance of the `OracleDate` structure and sets its value for date and time using the supplied `DateTime` value.

- OracleDate(string)

  This constructor creates a new instance of the `OracleDate` structure and sets its value using the supplied string.

- OracleDate(int, int, int)

  This constructor creates a new instance of the `OracleDate` structure and set its value for date using the supplied year, month, and day.

- OracleDate(int, int, int, int, int, int)

  This constructor creates a new instance of the `OracleDate` structure and set its value for time using the supplied year, month, day, hour, minute, and second.

- OracleDate(byte [ ])

  This constructor creates a new instance of the `OracleDate` structure and sets its value to the provided byte array, which is in the internal Oracle `DATE` format.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

OracleDate(DateTime)

This constructor creates a new instance of the `OracleDate` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#
public OracleDate (DateTime dt);
```

Parameters

- *dt*

  The provided `DateTime` value.

Remarks

The `OracleDate` structure only supports up to a second precision. The time value in the provided `DateTime` structure that has a precision smaller than second is ignored.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

OracleDate(string)

This constructor creates a new instance of the `OracleDate` structure and sets its value using the supplied string.

Declaration

```
// C#
public OracleDate (string dateStr);
```

Parameters

- *dateStr*

  A string that represents an Oracle `DATE`.

Exceptions

`ArgumentException` - The *dateStr* is an invalid string representation of an Oracle `DATE` or the *dateStr* is not in the date format specified by the thread's `OracleGlobalization.DateFormat` property, which represents Oracle's `NLS_DATE_FORMAT` parameter.

`ArgumentNullException` - The *dateStr* is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
```

```
// Set the nls_date_format for the Parse() method
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.DateFormat = "YYYY-MON-DD";
OracleGlobalization.SetThreadInfo(og);

// construct OracleDate from a string using the DateFormat specified.
OracleDate od = new OracleDate("1999-NOV-11");

// Set the nls_date_format for the OracleDate(string) constructor
og.DateFormat = "DD-MON-YYYY";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(od.ToString()); // Prints 11-NOV-1999
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39
- *Oracle Database SQL Reference* for further information on date format elements

OracleDate(int, int, int)

This constructor creates a new instance of the `OracleDate` structure and set its value for date using the supplied year, month, and day.

Declaration

```
// C#
public OracleDate (int year, int month, int day);
```

Parameters

- *year*

  The supplied year. Range of *year* is (-4712 to 9999).

- *month*

  The supplied month. Range of *month* is (1 to 12).

- *day*

  The supplied day. Range of *day* is (1 to 31).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleDate` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

OracleDate(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleDate` structure and set its value for time using the supplied year, month, day, hour, minute, and second.

Declaration

```
// C#
public OracleDate (int year, int month, int day, int hour, int minute, int
second);
```

Parameters

- *year*

  The supplied year. Range of *year* is (-4712 to 9999).

- *month*

  The supplied month. Range of *month* is (1 to 12).

- *day*

  The supplied day. Range of *day* is (1 to 31).

- *hour*

  The supplied hour. Range of *hour* is (0 to 23).

- *minute*

The supplied minute. Range of *minute* is (0 to 59).

- *second*

  The supplied second. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleDate` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

OracleDate(byte [ ])

This constructor creates a new instance of the `OracleDate` structure and sets its value to the provided byte array, which is in the internal Oracle `DATE` format.

Declaration

```
// C#
public OracleDate(byte [] bytes);
```

Parameters

- *bytes*

  A byte array that represents Oracle `DATE` in the internal Oracle `DATE` format.

Exceptions

`ArgumentException` - *bytes* is null or *bytes* is not in internal Oracle `DATE` format or `bytes` is not a valid Oracle `DATE`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

OracleDate Static Fields

The `OracleDate` static fields are listed in Table 5–21.

Table 5–21 OracleDate Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleDate` structure, which is December 31, 9999 23:59:59
MinValue	Represents the minimum valid date for an `OracleDate` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to the value of an `OracleDate` structure instance

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

MaxValue

This static field represents the maximum valid date for an `OracleDate` structure, which is December 31, 9999 23:59:59.

Declaration

```
// C#
public static readonly OracleDate MaxValue;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

MinValue

This static field represents the minimum valid date for an `OracleDate` structure, which is January 1, -4712.

Declaration

```
// C#
public static readonly OracleDate MinValue;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Null

This static field represents a null value that can be assigned to the value of an `OracleDate` instance.

Declaration

```
// C#
public static readonly OracleDate Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

OracleDate Static Methods

The `OracleDate` static methods are listed in Table 5–22.

Table 5–22 OracleDate Static Methods

Methods	Description
Equals	Determines if two `OracleDate` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleDate` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleDate` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleDate` values is less than the second

Table 5–22 OracleDate Static Methods (Cont.)

Methods	Description
LessThanOrEqual	Determines if the first of two `OracleDate` values is less than or equal to the second
NotEquals	Determines if two `OracleDate` values are not equal
GetSysDate	Returns an `OracleDate` structure that represents the current date and time
Parse	Returns an `OracleDate` structure and sets its value using a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Equals

Overloads `Object`

This method determines if two `OracleDate` values are equal.

Declaration

```
// C#
public static bool Equals(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First `OracleDate`.

- *value2*

  Second `OracleDate`.

Return Value

Returns `true` if two `OracleDate` values are equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

GreaterThan

This method determines if the first of two `OracleDate` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleDate value1, OracleDate value2);
```

Parameters

- *`value1`*

  First `OracleDate`.

- *`value2`*

  Second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

GreaterThanOrEqual

This method determines if the first of two `OracleDate` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First `OracleDate`.

- *value2*

  Second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

LessThan

This method determines if the first of two `OracleDate` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First `OracleDate`.

- *value2*

  Second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is less than the second. Otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

LessThanOrEqual

This method determines if the first of two `OracleDate` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First OracleDate.

- *value2*

  Second OracleDate.

Return Value

Returns true if the first of two OracleDate values is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDate that has a value compares greater than an OracleDate that has a null value.
- Two OracleDates that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

NotEquals

This method determines if two OracleDate values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First OracleDate.

- *value2*

  Second OracleDate.

Return Value

Returns `true` if two `OracleDate` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

GetSysDate

This method gets an `OracleDate` structure that represents the current date and time.

Declaration

```
// C#
public static OracleDate GetSysDate ();
```

Return Value

An `OracleDate` structure that represents the current date and time.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Parse

This method gets an `OracleDate` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#
```

```
public static OracleDate Parse (string dateStr);
```

Parameters

- *dateStr*

  A string that represents an Oracle DATE.

Return Value

An OracleDate structure.

Exceptions

ArgumentException - The *dateStr* is an invalid string representation of an Oracle DATE or the *dateStr* is not in the date format specified by the thread's OracleGlobalization.DateFormat property, which represents Oracle's NLS_DATE_FORMAT parameter.

ArgumentNullException - The *dateStr* is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_date_format for the Parse() method
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.DateFormat = "YYYY-MON-DD";
OracleGlobalization.SetThreadInfo(og);

// construct OracleDate from a string using the DateFormat specified.
OracleDate od = OracleDate.Parse("1999-NOV-11");

// Set the nls_date_format for the ToString() method
og.DateFormat = "DD-MON-YYYY";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(od.ToString()); // Prints 11-NOV-1999
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39
- *Oracle Database SQL Reference* for further information on datetime format elements

OracleDate Static Operators

The `OracleDate` static operators are listed in Table 5–23.

Table 5–23 OracleDate Static Operators

Operator	Description
operator ==	Determines if two `OracleDate` values are the same
operator >	Determines if the first of two `OracleDate` values is greater than the second
operator >=	Determines if the first of two `OracleDate` values is greater than or equal to the second
operator !=	Determines if the two `OracleDate` values are not equal
operator <	Determines if the first of two `OracleDate` values is less than the second
operator <=	Determines if the first of two `OracleDate` values is less than or equal to the second

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

operator ==

This method determines if two `OracleDate` values are the same.

Declaration

```
// C#
public static bool operator == (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First `OracleDate`.

- *value2*

  Second `OracleDate`.

Return Value

Returns `true` if they are the same; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

operator >

This method determines if the first of two `OracleDate` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First `OracleDate`.

- *value2*

  Second OracleDate.

Return Value

Returns true if the first of two OracleDate values is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDate that has a value compares greater than an OracleDate that has a null value.
- Two OracleDates that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

operator >=

This method determines if the first of two OracleDate values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First OracleDate.

- *value2*

  Second OracleDate.

Return Value

Returns true if the first of two OracleDate values is greater than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

operator !=

This method determines if the two `OracleDate` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First `OracleDate`.

- *value2*

  Second `OracleDate`.

Return Value

Returns `true` if the two `OracleDate` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

operator <

This method determines if the first of two `OracleDate` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First `OracleDate`.

- *value2*

  Second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

operator <=

This method determines if the first of two `OracleDate` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleDate value1, OracleDate value2);
```

Parameters

- *value1*

  First `OracleDate`.

- *value2*

  Second `OracleDate`.

Return Value

Returns `true` if the first of two `OracleDate` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

OracleDate Static Type Conversions

The `OracleDate` static type conversions are listed in Table 5–24.

Table 5–24 OracleDate Static Type Conversions

Operator	Description
explicit operator DateTime	Converts a structure to a `DateTime` structure
explicit operator OracleDate	Converts a structure to an `OracleDate` structure (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

explicit operator DateTime

This method converts an `OracleDate` structure to a `DateTime` structure.

Declaration

```
// C#
public static explicit operator DateTime(OracleDate val);
```

Parameters

- *val*

  An `OracleDate` structure.

Return Value

A `DateTime` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

explicit operator OracleDate

`explicit operator OracleDate` converts the provided structure to a `OracleDate` structure.

Overload List:

- explicit operator OracleDate(DateTime)

  This method converts a `DateTime` structure to an `OracleDate` structure.

- explicit operator OracleDate(OracleTimeStamp)

  This method converts an `OracleTimeStamp` structure to an `OracleDate` structure.

- explicit operator OracleDate(string)

  This method converts the supplied string to an `OracleDate` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

explicit operator OracleDate(DateTime)

This method converts a `DateTime` structure to an `OracleDate` structure.

Declaration

```
// C#
public static explicit operator OracleDate(DateTime dt);
```

Parameters

- *dt*

  A `DateTime` structure.

Return Value

An `OracleDate` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

explicit operator OracleDate(OracleTimeStamp)

This method converts an `OracleTimeStamp` structure to an `OracleDate` structure.

Declaration

```
// C#
public explicit operator OracleDate(OracleTimeStamp ts);
```

Parameters

- *ts*

  `OracleTimeStamp`

Return Value

The returned `OracleDate` structure contains the date and time in the `OracleTimeStamp` structure.

Remarks

The precision of the `OracleTimeStamp` value can be lost during the conversion.

If the `OracleTimeStamp` structure has a null value, the returned `OracleDate` structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

explicit operator OracleDate(string)

This method converts the supplied string to an `OracleDate` structure.

Declaration

```
// C#
public explicit operator OracleDate (string dateStr);
```

Parameters

- *dateStr*

  A string representation of an Oracle `DATE`.

Return Value

The returned `OracleDate` structure contains the date and time in the string *`dateStr`*.

Exceptions

`ArgumentNullException` - The `dateStr` is null.

`ArgumentException` - This exception is thrown if any of the following conditions exist:

- The *`dateStr`* is an invalid string representation of an Oracle `DATE`.
- The *`dateStr`* is not in the date format specified by the thread's `OracleGlobalization.DateFormat` property, which represents Oracle's `NLS_DATE_FORMAT` parameter.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_date_format for the Parse() method
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.DateFormat = "YYYY-MON-DD";
OracleGlobalization.SetThreadInfo(og);

// construct OracleDate from a string using the DateFormat specified.
OracleDate od = (OracleDate) "1999-NOV-11";

// Set the nls_date_format for the ToString() method
og.DateFormat = "DD-MON-YYYY";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(od.ToString()); // Prints 11-NOV-1999
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

OracleDate Properties

The `OracleDate` properties are listed in Table 5–25.

Table 5–25 OracleDate Properties

Properties	Description
BinData	Gets an array of bytes that represents an Oracle `DATE` in Oracle internal format
Day	Gets the day component of an `OracleDate` method
IsNull	Indicates whether the current instance has a null value
Hour	Gets the `hour` component of an `OracleDate`
Minute	Gets the minute component of an `OracleDate`
Month	Gets the `month` component of an `OracleDate`
Second	Gets the `second` component of an `OracleDate`
Value	Gets the date and time that is stored in the `OracleDate` structure
Year	Gets the `year` component of an `OracleDate`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

BinData

This property gets a array of bytes that represents an Oracle `DATE` in Oracle internal format.

Declaration

```
// C#
public byte[] BinData{get;}
```

Property Value

An array of bytes.

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Day

This property gets the day component of an `OracleDate`.

Declaration

```
// C#
public int Day{get;}
```

Property Value

A number that represents the day. Range of `Day` is (1 to 31).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

IsNull

This property indicates whether the current instance has a null value.

Declaration

```
// C#
public bool IsNull{get;}
```

Property Value

Returns `true` if the current instance has a null value; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Hour

This property gets the `hour` component of an `OracleDate`.

Declaration

```
// C#
public int Hour {get;}
```

Property Value

A number that represents `Hour`. Range of `Hour` is (0 to 23).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Minute

This property gets the minute component of an `OracleDate`.

Declaration

```
// C#
public int Minute {get;}
```

Property Value

A number that represents `Minute`. Range of `Minute` is (0 to 59).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Month

This property gets the `month` component of an `OracleDate`.

Declaration

```
// C#
public int Month {get;}
```

Property Value

A number that represents `Month`. Range of `Month` is (1 to 12).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Second

This property gets the `second` component of an `OracleDate`.

Declaration

```
// C#
public int Second {get;}
```

Property Value

A number that represents `Second`. Range of `Second` is (0 to 59).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Value

This property specifies the date and time that is stored in the `OracleDate` structure.

Declaration

```
// C#
public DateTime Value {get;}
```

Property Value

A `DateTime`.

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Year

This property gets the `year` component of an `OracleDate`.

Declaration

```
// C#
public int Year {get;}
```

Property Value

A number that represents `Year`. Range of `Year` is (-4712 to 9999).

Exceptions

`OracleNullValueException` - `OracleDate` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

OracleDate Methods

The `OracleDate` methods are listed in Table 5–26.

Table 5–26 OracleDate Methods

Methods	Description
CompareTo	Compares the current `OracleDate` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether an object has the same date and time as the current `OracleDate` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleDate` instance
GetDaysBetween	Calculates the number of days between the current `OracleDate` instance and an `OracleDate` structure
GetType	Inherited from `Object`
ToOracleTimeStamp	Converts the current `OracleDate` structure to an `OracleTimeStamp` structure
ToString	Converts the current `OracleDate` structure to a `string`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

CompareTo

This method compares the current `OracleDate` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  An object.

Return Value

The method returns:

- Less than zero: if the current `OracleDate` instance value is less than that of *obj*.
- Zero: if the current `OracleDate` instance and *obj* values are equal.
- Greater than zero: if the current `OracleDate` instance value is greater than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *obj* parameter is not an instance of `OracleDate`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleDates`. For example, comparing an `OracleDate` instance with an `OracleBinary` instance is not allowed. When an `OracleDate` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

Equals

This method determines whether an object has the same date and time as the current `OracleDate` instance.

Declaration

```
// C#
public override bool Equals( object obj);
```

Parameters

- *obj*

  An object.

Return Value

Returns `true` if *obj* has the same type as the current instance and represents the same date and time; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDate` that has a value compares greater than an `OracleDate` that has a null value.
- Two `OracleDates` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleDate` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

GetDaysBetween

This method calculates the number of days between the current `OracleDate` instance and the supplied `OracleDate` structure.

Declaration

```
// C#
public int GetDaysBetween (OracleDate val);
```

Parameters

- *`val`*

  An `OracleDate` structure.

Return Value

The number of days between the current `OracleDate` instance and the `OracleDate` structure.

Exceptions

`OracleNullValueException` - The current instance or the supplied `OracleDate` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

ToOracleTimeStamp

This method converts the current `OracleDate` structure to an `OracleTimeStamp` structure.

Declaration

```
// C#
public OracleTimeStamp ToOracleTimeStamp();
```

Return Value

An `OracleTimeStamp` structure.

Remarks

The returned `OracleTimeStamp` structure has date and time in the current instance.

If the `OracleDate` instance has a null value, the returned `OracleTimeStamp` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members

ToString

Overrides `ValueType`

This method converts the current `OracleDate` structure to a `string`.

Declaration

```
// C#
public override string ToString();
```

Return Value

A string.

Remarks

The returned value is a string representation of the `OracleDate` in the format specified by the thread's `OracleGlobalization.DateFormat` property. The names and abbreviations used for months and days are in the language specified by the thread's `OracleGlobalization.DateLanguage` and `OracleGlobalization.Calendar` properties. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_date_format for the Parse() method
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.DateFormat = "YYYY-MON-DD";
OracleGlobalization.SetThreadInfo(og);

// construct OracleDate from a string using the DateFormat specified.
OracleDate od = new OracleDate("1999-NOV-11");

// Set the nls_date_format for the ToString() method
og.DateFormat = "DD-MON-YYYY";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(od.ToString()); // Prints 11-NOV-1999
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

OracleDecimal Structure

The OracleDecimal structure represents an Oracle NUMBER in the database or any Oracle numeric value.

Class Inheritance

```
Object
  ValueType
    OracleDecimal
```

Declaration

```
// C#
public struct OracleDecimal : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

OracleDecimal can store up to 38 precision, while the .NET Decimal datatype can only hold up to 28 precision. When accessing the OracleDecimal.Value property from an OracleDecimal that has a value greater than 28 precision, loss of precision can occur. To retrieve the actual value of OracleDecimal, use the OracleDecimal.ToString() method. Another approach is to obtain the OracleDecimal value as a byte array in an internal Oracle NUMBER format through the BinData property.

Example

```
// C#
// Illustrates the usage of OracleDecimal
OracleDecimal pi = OracleDecimal.Pi;
// Use of implicit operator OracleDecimal(int)
OracleDecimal approxPi = OracleDecimal.Divide(22,7);

// Round the difference to 5 decimal places
OracleDecimal diff=OracleDecimal.Round(OracleDecimal.Abs(pi -approxPi),5);

// Set the format for ToString() - display with trailing zeroes
diff.Format = "9.9999900";
```

```
Console.WriteLine(diff.ToString());
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Constructors
- OracleDecimal Static Fields
- OracleDecimal Static (Comparison) Methods
- OracleDecimal Static (Manipulation) Methods
- OracleDecimal Static (Logarithmic) Methods
- OracleDecimal Static (Trigonometric) Methods
- OracleDecimal Static (Comparison) Operators
- OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)
- OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)
- OracleDecimal Properties
- OracleDecimal Instance Methods

OracleDecimal Members

`OracleDecimal` members are listed in the following tables:

OracleDecimal Constructors

`OracleDecimal` constructors are listed in Table 5–27

Table 5–27 OracleDecimal Constructors

Constructor	Description
OracleDecimal Constructors	Instantiates a new instance of `OracleDecimal` structure (Overloaded)

OracleDecimal Static Fields

The `OracleDecimal` static fields are listed in Table 5–28.

Table 5–28 OracleDecimal Static Fields

Field	Description
MaxPrecision	A constant representing the maximum precision, which is 38
MaxScale	A constant representing the maximum scale, which is 127
MaxValue	A constant representing the maximum value for this structure, which is $9.9…9 \times 10^{125}$
MinScale	A constant representing the minimum scale, which is -84
MinValue	A constant representing the minimum value for this structure, which is -1.0×10^{130}
NegativeOne	A constant representing the negative one value
Null	Represents a null value that can be assigned to an `OracleDecimal` instance
One	A constant representing the positive one value
Pi	A constant representing the numeric Pi value
Zero	A constant representing the zero value

OracleDecimal Static (Comparison) Methods

The `OracleDecimal` static (comparison) methods are listed in Table 5–29.

Table 5–29 OracleDecimal Static (Comparison) Methods

Methods	Description
Equals	Determines if two `OracleDecimal` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleDecimal` values is greater than the second

Table 5–29 OracleDecimal Static (Comparison) Methods (Cont.)

Methods	Description
GreaterThanOrEqual	Determines if the first of two `OracleDecimal` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleDecimal` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleDecimal` values is less than or equal to the second.
NotEquals	Determines if two `OracleDecimal` values are not equal

OracleDecimal Static (Manipulation) Methods

The `OracleDecimal` static (manipulation) methods are listed in Table 5–30.

Table 5–30 OracleDecimal Static (Manipulation) Methods

Methods	Description
Abs	Returns the absolute value of an `OracleDecimal`
Add	Adds two `OracleDecimal` structures
AdjustScale	Returns a new `OracleDecimal` with the specified number of digits and indicates whether or not to round or truncate the number if the scale is less than original
Ceiling	Returns a new `OracleDecimal` structure with its value set to the ceiling of an `OracleDecimal` structure
ConvertToPrecScale	Returns a new `OracleDecimal` structure with a new precision and scale
Divide	Divides one `OracleDecimal` value by another
Floor	Returns a new `OracleDecimal` structure with its value set to the floor of an `OracleDecimal` structure
Max	Returns the maximum value of the two supplied `OracleDecimal` structures
Min	Returns the minimum value of the two supplied `OracleDecimal` structures
Mod	Returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures
Multiply	Returns a new `OracleDecimal` structure with its value set to the result of multiplying two `OracleDecimal` structures

Table 5–30 OracleDecimal Static (Manipulation) Methods (Cont.)

Methods	Description
Negate	Returns a new `OracleDecimal` structure with its value set to the negation of the supplied `OracleDecimal` structure
Parse	Converts a `string` to an `OracleDecimal`
Round	Returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure and rounded off to the specified place
SetPrecision	Returns a new `OracleDecimal` structure with a new specified precision.
Shift	Returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure, and its decimal place shifted to the specified number of places to the right
Sign	Determines the sign of an `OracleDecimal` structure
Sqrt	Returns a new `OracleDecimal` structure with its value set to the square root of the supplied `OracleDecimal` structure
Subtract	Returns a new `OracleDecimal` structure with its value set to result of subtracting one `OracleDecimal` structure from another
Truncate	Truncates the `OracleDecimal` at a specified position

OracleDecimal Static (Logarithmic) Methods

The `OracleDecimal` static (logarithmic) methods are listed in Table 5–31.

Table 5–31 OracleDecimal Static (Logarithmic) Methods

Methods	Description
Exp	Returns a new `OracleDecimal` structure with its value set to e raised to the supplied power
Log	Returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure (Overloaded)
Pow	Returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` structure raised to the supplied power (Overloaded)

OracleDecimal Static (Trigonometric) Methods

The `OracleDecimal` static (trigonometric) methods are listed in Table 5–32.

Table 5–32 OracleDecimal Static (Trigonometric) Methods

Methods	Description
Acos	Returns an angle in radian whose cosine is the supplied `OracleDecimal` structure
Asin	Returns an angle in radian whose sine is the supplied `OracleDecimal` structure
Atan	Returns an angle in radian whose tangent is the supplied `OracleDecimal` structure
Atan2	Returns an angle in radian whose tangent is the quotient of the two supplied `OracleDecimal` structures
Cos	Returns the cosine of the supplied angle in radian
Sin	Returns the sine of the supplied angle in radian
Tan	Returns the tangent of the supplied angle in radian
Cosh	Returns the hyperbolic cosine of the supplied angle in radian
Sinh	Returns the hyperbolic sine of the supplied angle in radian
Tanh	Returns the hyperbolic tangent of the supplied angle in radian

OracleDecimal Static (Comparison) Operators

The `OracleDecimal` static (comparison) operators are listed in Table 5–33.

Table 5–33 OracleDecimal Static (Comparison) Operators

Operator	Description
operator +	Adds two `OracleDecimal` values
operator /	Divides one `OracleDecimal` value by another
operator ==	Determines if the two `OracleDecimal` values are equal
operator >	Determines if the first of two `OracleDecimal` values is greater than the second
operator >=	Determines if the first of two `OracleDecimal` values is greater than or equal to the second
operator !=	Determines if the two `OracleDecimal` values are not equal

Table 5–33 OracleDecimal Static (Comparison) Operators (Cont.)

Operator	Description
operator <	Determines if the first of two `OracleDecimal` values is less than the second
operator <=	Determines if the first of two `OracleDecimal` values is less than or equal to the second
operator *	Multiplies two `OracleDecimal` structures
operator -	Subtracts one `OracleDecimal` structure from another
operator -	Negates an `OracleDecimal` structure
operator%	Returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures.

OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

The `OracleDecimal` static operators (Conversion from .NET Type to `OracleDecimal`) are listed in Table 5–34.

Table 5–34 OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

Operator	Description
implicit operator OracleDecimal	Converts an instance value to an `OracleDecimal` structure (Overloaded)
explicit operator OracleDecimal	Converts an instance value to an `OracleDecimal` structure (Overloaded)

OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

The `OracleDecimal` static operators (Conversion from `OracleDecimal` to .NET) are listed in Table 5–35.

Table 5–35 OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

Operator	Description
explicit operator byte	Returns the `byte` representation of the `OracleDecimal` value

Table 5–35 OracleDecimal Static Operators (Conversion from OracleDecimal to .NET) (Cont.)

Operator	Description
explicit operator decimal	Returns the `decimal` representation of the `OracleDecimal` value
explicit operator double	Returns the `double` representation of the `OracleDecimal` value
explicit operator short	Returns the `short` representation of the `OracleDecimal` value
explicit operator int	Returns the `int` representation of the `OracleDecimal` value
explicit operator long	Returns the `long` representation of the `OracleDecimal` value
explicit operator float	Returns the `float` representation of the `OracleDecimal` value

OracleDecimal Properties

The `OracleDecimal` properties are listed in Table 5–36.

Table 5–36 OracleDecimal Properties

Properties	Description
BinData	Returns a byte array that represents the Oracle `NUMBER` in Oracle internal format
Format	Specifies the format for `ToString()`
IsInt	Indicates whether the current instance is an integer
IsNull	Indicates whether the current instance has a null value
IsPositive	Indicates whether the current instance is greater than `0`
IsZero	Indicates whether the current instance has a `zero` value
Value	Returns a `decimal` value

OracleDecimal Instance Methods

The `OracleDecimal` instance methods are listed in Table 5–37.

Table 5–37 OracleDecimal Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether an object is an instance of `OracleDecimal`, and whether the value of the object is equal to the current instance (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from `Object`
ToByte	Returns the `byte` representation of the current instance
ToDouble	Returns the `double` representation of the current instance
ToInt16	Returns the `Int16` representation of the current instance
ToInt32	Returns the `Int32` representation of the current instance
ToInt64	Returns the `Int64` representation of the current instance
ToSingle	Returns the `Single` representation of the current instance
ToString	Overloads `Object.ToString()` Returns the `string` representation of the current instance

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Structure

OracleDecimal Constructors

The `OracleDecimal` constructors instantiate a new instance of the `OracleDecimal` structure.

Overload List:

- OracleDecimal(byte [ ])

  This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied byte array, which is in an Oracle `NUMBER` format.

- OracleDecimal(decimal)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Decimal` value.

- OracleDecimal(double)

  This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `double` value.

- OracleDecimal(int)

  This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Int32` value.

- OracleDecimal(float)

  This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Single` value.

- OracleDecimal(long)

  This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Int64` value.

- OracleDecimal(string)

  This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `string` value.

- OracleDecimal(string, string)

  This constructor creates a new instance of the `OracleDecimal` structure with the supplied `string` value and number format.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal(byte [ ])

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied byte array, which is in an Oracle `NUMBER` format.

Declaration

```
// C#
public OracleDecimal(byte [] bytes);
```

Parameters

- *bytes*

  A byte array that represents an Oracle NUMBER in an internal Oracle format.

Exceptions

ArgumentException - The *bytes* parameter is not in a internal Oracle NUMBER format or *bytes* has an invalid value.

ArgumentNullException - The *bytes* parameter is null.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal(decimal)

This constructor creates a new instance of the OracleDecimal structure and sets its value to the supplied Decimal value.

Declaration

```
// C#
public OracleDecimal(decimal decX);
```

Parameters

- *decX*

  The provided Decimal value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal(double)

This constructor creates a new instance of the OracleDecimal structure and sets its value to the supplied double value.

Declaration

```
// C#
public OracleDecimal(double doubleX)
```

Parameters

- *doubleX*

  The provided double value.

Exceptions

`OverFlowException` - The value of the supplied `double` is greater than the maximum value or less than the minimum value of `OracleDecimal`.

Remarks

`OracleDecimal` contains the following values depending on the provided double value:

- `double.PositiveInfinity`: positive infinity value
- `double.NegativeInfinity`: negative infinity value.
- `double.NaN`: null value

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal(int)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Int32` value.

Declaration

```
// C#
public OracleDecimal(int intX);
```

Parameters

- *intX*

  The provided `Int32` value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal(float)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Single` value.

Declaration

```
// C#
public OracleDecimal(float floatX);
```

Parameters

- *`floatX`*

  The provided `float` value.

Remarks

`OracleDecimal` contains the following values depending on the provided `float` value:

`float.PositiveInfinity`: positive infinity value

`float.NegativeInfinity`: negative infinity value

`float.NaN`: null value

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal(long)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Int64` value.

Declaration

```
// C#
public OracleDecimal(long longX);
```

Parameters

- *longX*

  The provided `Int64` value.

  See Also:

  - Oracle.DataAccess.Types Namespace (ODP.NET Types)
  - OracleDecimal Members
  - OracleDecimal Structure

OracleDecimal(string)

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `string` value.

Declaration

```
// C#
public OracleDecimal(string numStr);
```

Parameters

- *numStr*

  The provided `string` value.

Exceptions

`ArgumentException` - The *numStr* parameter is an invalid string representation of an `OracleDecimal`.

`ArgumentNullException` - The *numStr* parameter is null.

`OverFlowException` - The value of *numStr* is greater than the maximum value or less than the minimum value of `OracleDecimal`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

OracleDecimal(string, string)

This constructor creates a new instance of the `OracleDecimal` structure with the supplied `string` value and number format.

Declaration

```
// C#
public OracleDecimal(string numStr, string format);
```

Parameters

- *`numStr`*

  The provided `string` value.

- *`format`*

  The provided number `format`.

Exceptions

`ArgumentException` - The *`numStr`* parameter is an invalid string representation of an `OracleDecimal` or the *`numStr`* is not in the numeric format specified by *`format`*.

`ArgumentNullException` - The *`numStr`* parameter is null.

`OverFlowException` - The value of *`numStr`* parameter is greater than the maximum value or less than the minimum value of `OracleDecimal`.

Remarks

If the numeric format includes decimal and group separators, then the provided string must use those characters defined by the `OracleGlobalization.NumericCharacters` of the thread.

If the numeric format includes the currency symbol, ISO currency symbol, or the dual currency symbol, then the provided string must use those symbols defined by the `OracleGlobalization.Currency`, `OracleGlobalization.ISOCurrency`, and `OracleGlobalization.DualCurrency` properties respectively.

Example

```
// C#
// Set the nls parameters to be used in the numeric format
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.Currency = "$";
og.NumericCharacters = ".,";
OracleGlobalization.SetThreadInfo(og);

// Construct an OracleDecimal using a valid numeric format
OracleDecimal od = new OracleDecimal("$2,222.22","L9G999D99");
Console.WriteLine(od.ToString()); // Prints $2,222.22
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

OracleDecimal Static Fields

The `OracleDecimal` static fields are listed in Table 5–38.

Table 5–38 OracleDecimal Static Fields

Field	Description
MaxPrecision	A constant representing the maximum precision, which is 38
MaxScale	A constant representing the maximum scale, which is 127
MaxValue	A constant representing the maximum value for this structure, which is $9.9\ldots9 \times 10^{125}$
MinScale	A constant representing the minimum scale, which is -84

Table 5–38 OracleDecimal Static Fields (Cont.)

Field	Description
MinValue	A constant representing the minimum value for this structure, which is -1.0×10^{130}
NegativeOne	A constant representing the negative one value
Null	Represents a null value that can be assigned to an `OracleDecimal` instance
One	A constant representing the positive one value
Pi	A constant representing the numeric Pi value
Zero	A constant representing the zero value

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

MaxPrecision

This static field represents the maximum precision, which is 38.

Declaration

```
// C#
public static readonly byte MaxPrecision;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

MaxScale

This static field a constant representing the maximum scale, which is 127.

Declaration

```
// C#
public static readonly byte MaxScale;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

MaxValue

This static field indicates a constant representing the maximum value for this structure, which is 9.9…9 x 10^{125} (38 nines followed by 88 zeroes).

Declaration

```
// C#
public static readonly OracleDecimal MaxValue;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

MinScale

This static field indicates a constant representing the maximum scale, which is -84.

Declaration

```
// C#
public static readonly int MinScale;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

MinValue

This static field indicates a constant representing the minimum value for this structure, which is -1.0 x 10^{130}.

Declaration

```
// C#
public static readonly OracleDecimal MinValue;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

NegativeOne

This static field indicates a constant representing the negative one value.

Declaration

```
// C#
public static readonly OracleDecimal NegativeOne;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Null

This static field represents a null value that can be assigned to an `OracleDecimal` instance.

Declaration

```
// C#
public static readonly OracleDecimal Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

One

This static field indicates a constant representing the positive one value.

Declaration

```
// C#
public static readonly OracleDecimal One;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Pi

This static field indicates a constant representing the numeric Pi value.

Declaration

```
// C#
public static readonly OracleDecimal Pi;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Zero

This static field indicates a constant representing the zero value.

Declaration

```
// C#
public static readonly OracleDecimal Zero;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal Static (Comparison) Methods

The `OracleDecimal` static (comparison) methods are listed in Table 5–39.

Table 5–39 OracleDecimal Static (Comparison) Methods

Methods	Description
Equals	Determines if two `OracleDecimal` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleDecimal` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleDecimal` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleDecimal` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleDecimal` values is less than or equal to the second.
NotEquals	Determines if two `OracleDecimal` values are not equal

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Equals

This method determines if two `OracleDecimal` values are equal.

Declaration

```
// C#
public static bool Equals(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*

  First `OracleDecimal`.

- *value2*

  Second `OracleDecimal`.

Return Value

Returns `true` if two `OracleDecimal` values are equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

GreaterThan

This method determines if the first of two `OracleDecimal` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*

  First `OracleDecimal`.

- *value2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

GreaterThanOrEqual

This method determines if the first of two `OracleDecimal` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleDecimal value1, OracleDecimal
value2);
```

Parameters

- *value1*

  First `OracleDecimal`.

- *value2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

LessThan

This method determines if the first of two `OracleDecimal` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*

  First `OracleDecimal`.

- *value2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

LessThanOrEqual

This method determines if the first of two `OracleDecimal` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*

  First `OracleDecimal`.

- *value2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

NotEquals

This method determines if two `OracleDecimal` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleDecimal value1, OracleDecimal value2);
```

Parameters

- *value1*

  First `OracleDecimal`.

- *value2*

  Second `OracleDecimal`.

Return Value

Returns `true` if two `OracleDecimal` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal Static (Manipulation) Methods

The `OracleDecimal` static (manipulation) methods are listed in Table 5–40.

Table 5–40 OracleDecimal Static (Manipulation) Methods

Methods	Description
Abs	Returns the absolute value of an `OracleDecimal`
Add	Adds two `OracleDecimal` structures
AdjustScale	Returns a new `OracleDecimal` with the specified number of digits and indicates whether or not to round or truncate the number if the scale is less than original
Ceiling	Returns a new `OracleDecimal` structure with its value set to the ceiling of an `OracleDecimal` structure
ConvertToPrecScale	Returns a new `OracleDecimal` structure with a new precision and scale
Divide	Divides one `OracleDecimal` value by another
Floor	Returns a new `OracleDecimal` structure with its value set to the floor of an `OracleDecimal` structure
Max	Returns the maximum value of the two supplied `OracleDecimal` structures

Table 5–40 OracleDecimal Static (Manipulation) Methods (Cont.)

Methods	Description
Min	Returns the minimum value of the two supplied `OracleDecimal` structures
Mod	Returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures
Multiply	Returns a new `OracleDecimal` structure with its value set to the result of multiplying two `OracleDecimal` structures
Negate	Returns a new `OracleDecimal` structure with its value set to the negation of the supplied `OracleDecimal` structure
Parse	Converts a `string` to an `OracleDecimal`
Round	Returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure and rounded off to the specified place
SetPrecision	Returns a new `OracleDecimal` structure with a new specified precision.
Shift	Returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure, and its decimal place shifted to the specified number of places to the right
Sign	Determines the sign of an `OracleDecimal` structure
Sqrt	Returns a new `OracleDecimal` structure with its value set to the square root of the supplied `OracleDecimal` structure
Subtract	Returns a new `OracleDecimal` structure with its value set to result of subtracting one `OracleDecimal` structure from another
Truncate	Truncates the `OracleDecimal` at a specified position

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Abs

This method returns the absolute value of an `OracleDecimal`.

Declaration

```
// C#
public static OracleDecimal Abs(OracleDecimal val);
```

Parameters

- *val*

  An OracleDecimal.

Return Value

The absolute value of an OracleDecimal.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Add

This method adds two OracleDecimal structures.

Declaration

```
// C#
public static OracleDecimal Add(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First OracleDecimal.

- *val2*

  Second OracleDecimal.

Return Value

Returns an OracleDecimal structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

AdjustScale

This method returns a new `OracleDecimal` with the specified number of digits and indicates whether or not to round or truncate the number if the scale is less than the original.

Declaration

```
// C#
public static OracleDecimal AdjustScale(OracleDecimal val, int digits,
     bool fRound);
```

Parameters

- *val*

  An `OracleDecimal`.

- *digits*

  The number of digits.

- *fRound*

  Indicates whether to round or truncate the number. Setting it to `true` rounds the number and setting it to `false` truncates the number.

Return Value

An `OracleDecimal`.

Remarks

If the supplied `OracleDecimal` has a null value, the returned `OracleDecimal` has a null value.

Example

```
// C#
OracleDecimal od = new OracleDecimal(5.555);
// Adjust Scale to 2 with rounding off
OracleDecimal odr = OracleDecimal.AdjustScale(od, 2, true);
Console.WriteLine(odr.ToString()); // Prints 5.56
// Adjust Scale to 2 with truncation
OracleDecimal odt = OracleDecimal.AdjustScale(od, 2, false);
Console.WriteLine(odt.ToString()); // Prints 5.55
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Ceiling

This method returns a new `OracleDecimal` structure with its value set to the ceiling of the supplied `OracleDecimal`.

Declaration

```
// C#
public static OracleDecimal Ceiling(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

ConvertToPrecScale

This method returns a new `OracleDecimal` structure with a new precision and scale.

Declaration

```
// C#
public static OracleDecimal ConvertToPrecScale(OracleDecimal val, int precision,
  int scale);
```

Parameters

- *val*

  An `OracleDecimal` structure.

- *precision*

  The precision. Range of precision is 1 to 38.

- *scale*

  The number of digits to the right of the decimal point. Range of scale is -84 to 127.

Return Value

A new `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` has a null value, the returned `OracleDecimal` has a null value.

Example

```
// C#
OracleDecimal od = new OracleDecimal(555.6666);

// Set the precision of od to 5 and scale to 2
OracleDecimal odp5s2 = OracleDecimal.ConvertToPrecScale(od,5,2);
```

```
Console.WriteLine(odp5s2.ToString()); // Prints 555.67

// Set the precision of od to 3 and scale to 0
OracleDecimal odp3s0 = OracleDecimal.ConvertToPrecScale(od,3,0);
Console.WriteLine(odp3s0.ToString()); // Prints 556
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Divide

This method divides one `OracleDecimal` value by another.

Declaration

```
// C#
public static OracleDecimal Divide(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  An `OracleDecimal`.

- *val2*

  An `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Floor

This method returns a new `OracleDecimal` structure with its value set to the floor of the supplied `OracleDecimal` structure.

Declaration

```
// C#
public static OracleDecimal Floor(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Max

This method returns the maximum value of the two supplied `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal Max(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  An `OracleDecimal` structure.

- *val2*

  An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure that has the greater value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Min

This method returns the minimum value of the two supplied `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal Min(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  An `OracleDecimal` structure.

- *val2*

  An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure that has the smaller value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Mod

This method returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal Mod(OracleDecimal val1, OracleDecimal divider);
```

Parameters

- *val1*

  An `OracleDecimal` structure.

- *divider*

  An `OracleDecimal` structure.

Return Value

An `OracleDecimal`.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Multiply

This method returns a new `OracleDecimal` structure with its value set to the result of multiplying two `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal Multiply(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  An `OracleDecimal` structure.

- *val2*

  An `OracleDecimal` structure.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Negate

This method returns a new `OracleDecimal` structure with its value set to the negation of the supplied `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal Negate(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

A new `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Parse

This method converts a `string` to an `OracleDecimal`.

Declaration

```
// C#
public static OracleDecimal Parse (string str);
```

Parameters

- *str*

  The string being converted.

Return Value

A new `OracleDecimal` structure.

Exceptions

`ArgumentException` - The *numStr* parameter is an invalid string representation of an `OracleDecimal`.

`ArgumentNullException` - The *numStr* parameter is null.

`OverFlowException` - The value of *numStr* is greater than the maximum value or less than the minimum value of `OracleDecimal`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

Round

This method returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure and rounded off to the specified place.

Declaration

```
// C#
public static OracleDecimal Round(OracleDecimal val, int decplace);
```

Parameters

- *val*

An `OracleDecimal` structure.

- *decplace*

  The specified decimal place. If the value is positive, the function rounds the `OracleDecimal` structure to the right of the decimal point. If the value is negative, the function rounds to the left of the decimal point.

Return Value

An `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` structure has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

SetPrecision

This method returns a new `OracleDecimal` structure with a new specified precision.

Declaration

```
// C#
public static OracleDecimal SetPrecision(OracleDecimal val, int precision);
```

Parameters

- *val*

  An `OracleDecimal` structure.

- *precision*

  The specified precision. Range of precision is 1 to 38.

Return Value

An `OracleDecimal` structure.

Remarks

The returned `OracleDecimal` is rounded off if the specified precision is smaller than the precision of *`val`*.

If *`val`* has a null value, the returned `OracleDecimal` has a null value.

Example

```
// C#
OracleDecimal od = new OracleDecimal(555.6666);

// Set the precision of od to 3
OracleDecimal odp3 = OracleDecimal.SetPrecision(od,3);
Console.WriteLine(odp3.ToString()); // Prints 556

// Set the precision of od to 4
OracleDecimal odp4 = OracleDecimal.SetPrecision(od,4);
Console.WriteLine(odp4.ToString()); // Prints 555.7
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Shift

This method returns a new `OracleDecimal` structure with its value set to that of the supplied `OracleDecimal` structure, and its decimal place shifted to the specified number of places to the right.

Declaration

```
// C#
public static OracleDecimal Shift(OracleDecimal val, int decplaces);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.

- *`decplaces`*

  The specified number of places to be shifted.

Return Value

An `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` structure has a null value, the returned `OracleDecimal` has a null value.

If *`decplaces`* is negative, the shift is to the left.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Sign

This method determines the sign of an `OracleDecimal` structure.

Declaration

```
// C#
public static int Sign(OracleDecimal val);
```

Parameters

- *`val`*

  An `OracleDecimal` structure.

Return Value

- `-1`: if the supplied `OracleDecimal` < 0
- `0`: if the supplied `OracleDecimal` == 0
- `1`: if the supplied `OracleDecimal` > 0

Exceptions

`OracleNullValueException` - The argument has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Sqrt

This method returns a new `OracleDecimal` structure with its value set to the square root of the supplied `OracleDecimal` structure.

Declaration

```
// C#
public static OracleDecimal Sqrt(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure.

Exceptions

`ArgumentOutOfRangeException` - The provided `OracleDecimal` structure is less than zero.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Subtract

This method returns a new `OracleDecimal` structure with its value set to result of subtracting one `OracleDecimal` structure from another.

Declaration

```
// C#
public static OracleDecimal Subtract(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  An `OracleDecimal` structure.

- *val2*

  An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Truncate

This method truncates the `OracleDecimal` at a specified position.

Declaration

```
// C#
public static OracleDecimal Truncate(OracleDecimal val, int pos);
```

Parameters

- *val*

  An `OracleDecimal` structure.

- *pos*

  The specified position. If the value is positive, the function truncates the `OracleDecimal` structure to the right of the decimal point. If the value is

negative, it truncates the `OracleDecimal` structure to the left of the decimal point.

Return Value

An `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` structure has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal Static (Logarithmic) Methods

The `OracleDecimal` static (logarithmic) methods are listed in Table 5–41.

Table 5–41 OracleDecimal Static (Logarithmic) Methods

Methods	Description
Exp	Returns a new `OracleDecimal` structure with its value set to e raised to the supplied power
Log	Returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure (Overloaded)
Pow	Returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` structure raised to the supplied power (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Exp

This method returns a new `OracleDecimal` structure with its value set to e raised to the supplied `OracleDecimal`.

Declaration

```
// C#
public static OracleDecimal Exp(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Log

`Log` returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure.

Overload List:

- Log(OracleDecimal)

  This method returns a new `OracleDecimal` structure with its value set to the natural logarithm (base e) of the supplied `OracleDecimal` structure.

- Log(OracleDecimal, int)

  This method returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure in the supplied base.

- Log(OracleDecimal, OracleDecimal)

  This method returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure in the supplied base.

  See Also:

  - Oracle.DataAccess.Types Namespace (ODP.NET Types)
  - OracleDecimal Members
  - OracleDecimal Structure

Log(OracleDecimal)

This method returns a new `OracleDecimal` structure with its value set to the natural logarithm (base e) of the supplied `OracleDecimal` structure.

Declaration

```
// C#
public static OracleDecimal Log(OracleDecimal val);
```

Parameters

- *`val`*

  An `OracleDecimal` structure whose logarithm is to be calculated.

Return Value

Returns a new `OracleDecimal` structure with its value set to the natural logarithm (base e) of *`val`*.

Exceptions

`ArgumentOutOfRangeException` - The supplied `OracleDecimal` value is less than zero.

Remarks

If the supplied `OracleDecimal` structure has a null value, the returned `OracleDecimal` has a null value.

If the supplied `OracleDecimal` structure has zero value, the result is undefined, and the returned `OracleDecimal` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Log(OracleDecimal, int)

This method returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure in the supplied base.

Declaration

```
// C#
public static OracleDecimal Log(OracleDecimal val, int logBase);
```

Parameters

- *`val`*

  An `OracleDecimal` structure whose logarithm is to be calculated.

- *`logBase`*

  An `int` that specifies the base of the logarithm.

Return Value

A new `OracleDecimal` structure with its value set to the logarithm of *`val`* in the supplied base.

Exceptions

`ArgumentOutOfRangeException` - Either argument is less than zero.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

If both arguments have zero value, the result is undefined, and the returned `OracleDecimal` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Log(OracleDecimal, OracleDecimal)

This method returns the supplied `OracleDecimal` structure with its value set to the logarithm of the supplied `OracleDecimal` structure in the supplied base.

Declaration

```
// C#
public static OracleDecimal Log(OracleDecimal val, OracleDecimal logBase);
```

Parameters

- *`val`*

  An `OracleDecimal` structure whose logarithm is to be calculated.

- *`logBase`*

  An `OracleDecimal` structure that specifies the base of the logarithm.

Return Value

Returns the logarithm of *`val`* in the supplied base.

Exceptions

`ArgumentOutOfRangeException` - Either the *`val`* or *`logBase`* parameter is less than zero.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

If both arguments have zero value, the result is undefined, and the returned `OracleDecimal` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Pow

`Pow` returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` structure raised to the supplied power.

Overload List:

- Pow(OracleDecimal, int)

  This method returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` value raised to the supplied `Int32` power.

- Pow(OracleDecimal, OracleDecimal)

  This method returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` structure raised to the supplied `OracleDecimal` power.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Pow(OracleDecimal, int)

This method returns a new `OracleDecimal` structure with its value set to the supplied `OracleDecimal` value raised to the supplied `Int32` power.

Declaration

```
// C#
public static OracleDecimal Pow(OracleDecimal val, int power);
```

Parameters

- *val*

  An `OracleDecimal` structure.

- *power*

  An int value that specifies the power.

Return Value

An OracleDecimal structure.

Remarks

If the supplied OracleDecimal structure has a null value, the returned OracleDecimal has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Pow(OracleDecimal, OracleDecimal)

This method returns a new OracleDecimal structure with its value set to the supplied OracleDecimal structure raised to the supplied OracleDecimal power.

Declaration

```
// C#
public static OracleDecimal Pow(OracleDecimal val, OracleDecimal power);
```

Parameters

- *val*

  An OracleDecimal structure.

- *power*

  An OracleDecimal structure that specifies the power.

Return Value

An OracleDecimal structure.

Remarks

If the supplied OracleDecimal structure has a null value, the returned OracleDecimal has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal Static (Trigonometric) Methods

The `OracleDecimal` static (trigonometric) methods are listed in Table 5–42.

Table 5–42 OracleDecimal Static (Trigonometric) Methods

Methods	Description
Acos	Returns an angle in radian whose cosine is the supplied `OracleDecimal` structure
Asin	Returns an angle in radian whose sine is the supplied `OracleDecimal` structure
Atan	Returns an angle in radian whose tangent is the supplied `OracleDecimal` structure
Atan2	Returns an angle in radian whose tangent is the quotient of the two supplied `OracleDecimal` structures
Cos	Returns the cosine of the supplied angle in radian
Sin	Returns the sine of the supplied angle in radian
Tan	Returns the tangent of the supplied angle in radian
Cosh	Returns the hyperbolic cosine of the supplied angle in radian
Sinh	Returns the hyperbolic sine of the supplied angle in radian
Tanh	Returns the hyperbolic tangent of the supplied angle in radian

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Acos

This method returns an angle in radian whose cosine is the supplied `OracleDecimal` structure.

Declaration

```
// C#
public static OracleDecimal Acos(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure. Range is (-1 to 1).

Return Value

An `OracleDecimal` structure that represents an angle in radian.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Asin

This method returns an angle in radian whose sine is the supplied `OracleDecimal` structure.

Declaration

```
// C#
public static OracleDecimal Asin(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure. Range is (-1 to 1).

Return Value

An `OracleDecimal` structure that represents an angle in radian.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Atan

This method returns an angle in radian whose tangent is the supplied `OracleDecimal` structure

Declaration

```
// C#
public static OracleDecimal Atan(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal`.

Return Value

An `OracleDecimal` structure that represents an angle in radian.

Remarks

If the argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Atan2

This method returns an angle in radian whose tangent is the quotient of the two supplied `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal Atan2(OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  An `OracleDecimal` structure that represents the y-coordinate.

- *val2*

  An `OracleDecimal` structure that represents the x-coordinate.

Return Value

An `OracleDecimal` structure that represents an angle in radian.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Cos

This method returns the cosine of the supplied angle in radian.

Declaration

```
// C#
public static OracleDecimal Cos(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure that represents an angle in radian.

Return Value

An `OracleDecimal` instance.

Exceptions

`ArgumentOutOfRangeException` - The *val* parameter is positive or negative infinity.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Sin

This method returns the sine of the supplied angle in radian.

Declaration

```
// C#
public static OracleDecimal Sin(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

An `OracleDecimal` structure that represents an angle in radian.

Exceptions

`ArgumentOutOfRangeException` - The *val* parameter is positive or negative infinity.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Tan

This method returns the tangent of the supplied angle in radian.

Declaration

```
// C#
public static OracleDecimal Tan(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure that represents an angle in radian.

Return Value

An `OracleDecimal` instance.

Exceptions

`ArgumentOutOfRangeException` - The *val* parameter is positive or negative infinity.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Cosh

This method returns the hyperbolic cosine of the supplied angle in radian.

Declaration

```
// C#
public static OracleDecimal Cosh(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure that represents an angle in radian.

Return Value

An `OracleDecimal` instance.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Sinh

This method returns the hyperbolic sine of the supplied angle in radian.

Declaration

```
// C#
public static OracleDecimal Sinh(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure that represents an angle in radian.

Return Value

An `OracleDecimal` instance.

Remarks

If either argument has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Tanh

This method returns the hyperbolic tangent of the supplied angle in radian.

Declaration

```
// C#
public static OracleDecimal Tanh(OracleDecimal val);
```

Parameters

- *val*

  An OracleDecimal structure that represents an angle in radian.

Return Value

An OracleDecimal instance.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal Static (Comparison) Operators

The OracleDecimal static (comparison) operators are listed in Table 5–43.

Table 5–43 OracleDecimal Static (Comparison) Operators

Operator	Description
operator +	Adds two OracleDecimal values
operator /	Divides one OracleDecimal value by another
operator ==	Determines if the two OracleDecimal values are equal
operator >	Determines if the first of two OracleDecimal values is greater than the second
operator >=	Determines if the first of two OracleDecimal values is greater than or equal to the second
operator !=	Determines if the two OracleDecimal values are not equal
operator <	Determines if the first of two OracleDecimal values is less than the second
operator <=	Determines if the first of two OracleDecimal values is less than or equal to the second
operator *	Multiplies two OracleDecimal structures
operator -	Subtracts one OracleDecimal structure from another

Table 5–43 OracleDecimal Static (Comparison) Operators (Cont.)

Operator	Description
operator -	Negates an `OracleDecimal` structure
operator%	Returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator +

This method adds two `OracleDecimal` values.

Declaration

```
// C#
public static OracleDecimal operator + (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

An `OracleDecimal` structure.

Remarks

If either operand has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator /

This method divides one `OracleDecimal`value by another.

Declaration

```
// C#
public static OracleDecimal operator / (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

Second `OracleDecimal`.

Return Value

An `OracleDecimal` structure.

Remarks

If either operand has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator ==

This method determines if two `OracleDecimal` values are equal.

Declaration

```
// C#
public static bool operator == (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

Returns `true` if their values are equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator >

This method determines if the first of two `OracleDecimal` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the two `OracleDecimal` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator >=

This method determines if the first of two `OracleDecimal` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator !=

This method determines if the first of two `OracleDecimal` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the two `OracleDecimal` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator <

This method determines if the first of two `OracleDecimal` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleDecimal val1,  OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator <=

This method determines if the first of two `OracleDecimal` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

Returns `true` if the first of two `OracleDecimal` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator *

This method multiplies two `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal operator * (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If either operand has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator -

This method subtracts one `OracleDecimal` structure from another.

Declaration

```
// C#
public static OracleDecimal operator - (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- *val1*

  First `OracleDecimal`.

- *val2*

  Second `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If either operand has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator -

This method negates the supplied `OracleDecimal` structure.

Declaration

```
// C#
public static OracleDecimal operator - (OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If the supplied `OracleDecimal` structure has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

operator%

This method returns a new `OracleDecimal` structure with its value set to the modulus of two `OracleDecimal` structures.

Declaration

```
// C#
public static OracleDecimal operator % (OracleDecimal val,
     OracleDecimal divider);
```

Parameters

- *val*

  An `OracleDecimal`.

- *divider*

  An `OracleDecimal`.

Return Value

A new `OracleDecimal` structure.

Remarks

If either operand has a null value, the returned `OracleDecimal` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

The `OracleDecimal` static operators (Conversion from .NET Type to `OracleDecimal`) are listed in Table 5–44.

Table 5–44 OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

Operator	Description
implicit operator OracleDecimal	Converts an instance value to an `OracleDecimal` structure (Overloaded)
explicit operator OracleDecimal	Converts an instance value to an `OracleDecimal` structure (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

implicit operator OracleDecimal

`implicit operator OracleDecimal` returns the `OracleDecimal` representation of a value.

Overload List:

- implicit operator OracleDecimal(decimal)

  This method returns the `OracleDecimal` representation of a `decimal` value.

- implicit operator OracleDecimal(int)

  This method returns the `OracleDecimal` representation of an `int` value.

- implicit operator OracleDecimal(long)

  This method returns the `OracleDecimal` representation of a `long` value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

implicit operator OracleDecimal(decimal)

This method returns the `OracleDecimal` representation of a `decimal` value.

Declaration

```
// C#
public static implicit operator OracleDecimal(decimal val);
```

Parameters

- *val*

  A `decimal` value.

Return Value

An `OracleDecimal`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

implicit operator OracleDecimal(int)

This method returns the `OracleDecimal` representation of an `int` value.

Declaration

```
// C#
public static implicit operator OracleDecimal(int val);
```

Parameters

- *val*

  An `int` value.

Return Value

An `OracleDecimal`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

implicit operator OracleDecimal(long)

This method returns the `OracleDecimal` representation of a `long` value.

Declaration

```
// C#
public static implicit operator OracleDecimal(long val);
```

Parameters

- *val*

A `long` value.

Return Value

An `OracleDecimal`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator OracleDecimal

`OracleDecimal` returns the `OracleDecimal` representation of a value.

Overload List:

- explicit operator OracleDecimal(double)

  This method returns the `OracleDecimal` representation of a double.

- explicit operator OracleDecimal(string)

  This method returns the `OracleDecimal` representation of a string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator OracleDecimal(double)

This method returns the `OracleDecimal` representation of a double.

Declaration

```
// C#
public static explicit operator OracleDecimal(double val);
```

Parameters

- *val*

  A `double`.

Return Value

An `OracleDecimal`.

Exceptions

`OverFlowException` - The value of the supplied `double` is greater than the maximum value of `OracleDecimal` or less than the minimum value of `OracleDecimal`.

Remarks

`OracleDecimal` contains the following values depending on the provided double value:

- `double.PositiveInfinity`: positive infinity value
- `double.NegativeInfinity`: negative infinity value.
- `double.NaN`: null value

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator OracleDecimal(string)

This method returns the `OracleDecimal` representation of a string.

Declaration

```
// C#
public static explicit operator OracleDecimal(string numStr);
```

Parameters

- *`numStr`*

  A `string` that represents a numeric value.

Return Value

An `OracleDecimal`.

Exceptions

`ArgumentException` - The *`numStr`* parameter is an invalid string representation of an `OracleDecimal`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

The `OracleDecimal` static operators (Conversion from `OracleDecimal` to .NET) are listed in Table 5–45.

Table 5–45 OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

Operator	Description
explicit operator byte	Returns the `byte` representation of the `OracleDecimal` value
explicit operator decimal	Returns the `decimal` representation of the `OracleDecimal` value
explicit operator double	Returns the `double` representation of the `OracleDecimal` value
explicit operator short	Returns the `short` representation of the `OracleDecimal` value
explicit operator int	Returns the `int` representation of the `OracleDecimal` value
explicit operator long	Returns the `long` representation of the `OracleDecimal` value
explicit operator float	Returns the `float` representation of the `OracleDecimal` value

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator byte

This method returns the `byte` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator byte(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

A `byte`.

Exceptions

`OracleNullValueException` - `OracleDecimal` has a null value.

`OverFlowException`- The `byte` cannot represent the supplied `OracleDecimal` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator decimal

This method returns the `decimal` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator decimal(OracleDecimal val);
```

Parameters

- *val*

  An OracleDecimal structure.

Return Value

A decimal.

Exceptions

OracleNullValueException - The OracleDecimal has a null value.

OverFlowException - The decimal cannot represent the supplied OracleDecimal structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator double

This method returns the double representation of the OracleDecimal value.

Declaration

```
// C#
public static explicit operator double(OracleDecimal val);
```

Parameters

- *val*

  An OracleDecimal structure.

Return Value

A double.

Exceptions

OracleNullValueException - The OracleDecimal has a null value.

OverFlowException - The double cannot represent the supplied OracleDecimal structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator short

This method returns the `short` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator short(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

A `short`.

Exceptions

`OracleNullValueException` - The `OracleDecimal` has a null value.

`OverFlowException` - The `short` cannot represent the supplied `OracleDecimal` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator int

This method returns the `int` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator int(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

An `int`.

Exceptions

`OracleNullValueException` - The `OracleDecimal` has a null value.

`OverFlowException` - The `int` cannot represent the supplied `OracleDecimal` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator long

This method returns the `long` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator long(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

A `long`.

Exceptions

`OracleNullValueException` - The `OracleDecimal` has a null value.

`OverFlowException` - The `long` cannot represent the supplied `OracleDecimal` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

explicit operator float

This method returns the `float` representation of the `OracleDecimal` value.

Declaration

```
// C#
public static explicit operator float(OracleDecimal val);
```

Parameters

- *val*

  An `OracleDecimal` structure.

Return Value

A `float`.

Exceptions

`OracleNullValueException` - The `OracleDecimal` has a null value.

`OverFlowException` - The `float` cannot represent the supplied `OracleDecimal` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal Properties

The `OracleDecimal` properties are listed in Table 5–46.

Table 5–46 OracleDecimal Properties

Properties	Description
BinData	Returns a byte array that represents the Oracle `NUMBER` in Oracle internal format
Format	Specifies the format for `ToString()`
IsInt	Indicates whether the current instance is an integer
IsNull	Indicates whether the current instance has a null value
IsPositive	Indicates whether the current instance is greater than `0`
IsZero	Indicates whether the current instance has a `zero` value
Value	Returns a `decimal` value

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

BinData

This property returns a byte array that represents the Oracle `NUMBER` in an internal Oracle format.

Declaration

```
// C#
public byte[] BinData {get;}
```

Property Value

A byte array that represents the Oracle `NUMBER` in an internal Oracle format.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Format

This property specifies the format for `ToString()`.

Declaration

```
// C#
public string Format {get; set;}
```

Property Value

The string which specifies the format.

Remarks

`Format` is used when `ToString()` is called on an instance of an `OracleDecimal`. It is useful if the `ToString()` method needs a specific currency symbol, group, or decimal separator as part of a string.

By default, this property is `null` which indicates that no special formatting is used.

The decimal and group separator characters are specified by the thread's `OracleGlobalization.NumericCharacters`.

The currency symbols are specified by the following thread properties:

- `OracleGlobalization.Currency`
- `OracleGlobalization.ISOCurrency`
- `OracleGlobalization.DualCurrency`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

IsInt

This property indicates whether the current instance is an integer value.

Declaration

```
// C#
public bool IsInt {get;}
```

Property Value

A `bool` value that returns `true` if the current instance is an integer value; otherwise, returns `false`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

IsNull

This property indicates whether the current instance has a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

A `bool` value that returns `true` if the current instance has a null value; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

IsPositive

This property indicates whether the value of the current instance is greater than `0`.

Declaration

```
// C#
public bool IsPositive {get;}
```

Property Value

A `bool` value that returns `true` if the current instance is greater than `0`; otherwise, returns `false`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

IsZero

This property indicates whether the current instance has a zero value.

Declaration

```
// C#
public  bool IsZero{get;}
```

Property Value

A `bool` value that returns `true` if the current instance has a zero value; otherwise, returns `false`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Value

This method returns a `decimal` value.

Declaration

```
// C#
public decimal Value {get;}
```

Property Value

A `decimal` value.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`OverFlowException` - The `decimal` cannot represent the supplied `OracleDecimal` structure.

Remarks

Precision can be lost when the `decimal` value is obtained from an `OracleDecimal`. See Remarks under "OracleDecimal Structure" on page 5-71 for further information.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal Instance Methods

The `OracleDecimal` instance methods are listed in Table 5–47.

Table 5–47 OracleDecimal Instance Methods

Method	Description
CompareTo	Compares the current instance to the supplied object and returns an integer that represents their relative values
Equals	Determines whether an object is an instance of `OracleDecimal`, and whether the value of the object is equal to the current instance (Overloaded)
GetHashCode	Returns a hash code for the current instance
GetType	Inherited from `Object`
ToByte	Returns the `byte` representation of the current instance
ToDouble	Returns the `double` representation of the current instance
ToInt16	Returns the `Int16` representation of the current instance
ToInt32	Returns the `Int32` representation of the current instance
ToInt64	Returns the `Int64` representation of the current instance
ToSingle	Returns the `Single` representation of the current instance
ToString	Overloads `Object.ToString()` Returns the `string` representation of the current instance

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

CompareTo

This method compares the current instance to the supplied object and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

The supplied instance.

Return Value

The method returns a number:

- Less than zero: if the value of the current instance is less than *obj*.
- Zero: if the value of the current instance is equal to *obj*.
- Greater than zero: if the value of the current instance is greater than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The parameter is not of type `OracleDecimal`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleDecimals`. For example, comparing an `OracleDecimal` instance with an `OracleBinary` instance is not allowed. When an `OracleDecimal` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

Equals

Overrides `Object`

This method determines whether an object is an instance of `OracleDecimal`, and whether the value of the object is equal to the current instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *obj*

  An `OracleDecimal` instance.

Return Value

Returns `true` if *obj* is an instance of `OracleDecimal`, and the value of *obj* is equal to the current instance; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleDecimal` that has a value compares greater than an `OracleDecimal` that has a null value.
- Two `OracleDecimals` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

GetHashCode

Overrides `Object`

This method returns a hash code for the current instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

Returns a hash code.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

ToByte

This method returns the `byte` representation of the current instance.

Declaration

```
// C#
public byte ToByte();
```

Return Value

A `byte`.

Exceptions

`OverFlowException` - The `byte` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

ToDouble

This method returns the `double` representation of the current instance.

Declaration

```
// C#
public double ToDouble();
```

Return Value

A `double`.

Exceptions

`OverFlowException` - The `double` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

ToInt16

This method returns the `Int16` representation of the current instance.

Declaration

```
// C#
public short ToInt16();
```

Return Value

A `short`.

Exceptions

`OverFlowException` - The `short` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

ToInt32

This method returns the `Int32` representation of the current instance.

Declaration

```
// C#
public int ToInt32();
```

Return Value

An `int`.

Exceptions

`OverFlowException` - The `int` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

ToInt64

This method returns the `Int64` representation of the current instance.

Declaration

```
// C#
public long ToInt64();
```

Return Value

A `long`.

Exceptions

`OverFlowException` - The `long` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

ToSingle

This method returns the `Single` representation of the current instance.

Declaration

```
// C#
public float ToSingle();
```

Return Value

A `float`.

Exceptions

`OverFlowException` - The `float` cannot represent the current instance.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure

ToString

Overrides `Object`

This method returns the `string` representation of the current instance.

Declaration

```
// C#
public override string ToString();
```

Return Value

Returns the number in a string.

Remarks

If the current instance has a null value, the returned string is "null".

The returned value is a string representation of an `OracleDecimal` in the numeric format specified by the `Format` property.

The decimal and group separator characters are specified by the thread's `OracleGlobalization.NumericCharacters`.

The currency symbols are specified by the following thread properties:

- `OracleGlobalization.Currency`

- `OracleGlobalization.ISOCurrency`
- `OracleGlobalization.DualCurrency`

If the numeric format is not specified, an Oracle default value is used.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

OracleIntervalDS Structure

The `OracleIntervalDS` structure represents the Oracle `INTERVAL DAY TO SECOND` datatype to be stored in or retrieved from a database. Each `OracleIntervalDS` stores a period of time in term of days, hours, minutes, seconds, and fractional seconds.

Class Inheritance

```
Object

  ValueType

    OracleIntervalDS
```

Declaration

```
// C#
public struct OracleIntervalDS : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
//  Illustrates usage of OracleIntervalDS

OracleIntervalDS idsMax = OracleIntervalDS.MaxValue;
double maxDays = idsMax.TotalDays;
maxDays -= 1;
OracleIntervalDS idsMax_1 = new OracleIntervalDS(maxDays);

// Calculate the difference. It should be 1 +/- epsilon days
// where epsilon for OracleIntervalDS = 0.000000001 seconds.

OracleIntervalDS idsDiff = idsMax - idsMax_1;

// If the difference isnt exactly 1 day, display the difference
if (idsDiff.TotalDays != 1)
   Console.WriteLine(idsDiff.ToString());
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Members
- OracleIntervalDS Constructors
- OracleIntervalDS Static Fields
- OracleIntervalDS Static Methods
- OracleIntervalDS Static Operators
- OracleIntervalDS Type Conversions
- OracleIntervalDS Properties
- OracleIntervalDS Methods

OracleIntervalDS Members

`OracleIntervalDS` members are listed in the following tables:

OracleIntervalDS Constructors

`OracleIntervalDS` constructors are listed in Table 5–48

Table 5–48 OracleIntervalDS Constructors

Constructor	Description
OracleIntervalDS Constructors	Instantiates a new instance of `OracleIntervalDS` structure (Overloaded)

OracleIntervalDS Static Fields

The `OracleIntervalDS` static fields are listed in Table 5–49.

Table 5–49 OracleIntervalDS Static Fields

Field	Description
MaxValue	Represents the maximum valid time interval for an `OracleIntervalDS` structure
MinValue	Represents the minimum valid time interval for an `OracleIntervalDS` structure
Null	Represents a null value that can be assigned to an `OracleIntervalDS` instance
Zero	Represents a zero value for an `OracleIntervalDS` structure

OracleIntervalDS Static Methods

The `OracleIntervalDS` static methods are listed in Table 5–50.

Table 5–50 OracleIntervalDS Static Methods

Methods	Description
Equals	Determines whether two `OracleIntervalDS` values are equal (Overloaded)
GreaterThan	Determines whether one `OracleIntervalDS` value is greater than another
GreaterThanOrEqual	Determines whether one `OracleIntervalDS` value is greater than or equal to another
LessThan	Determines whether one `OracleIntervalDS` value is less than another
LessThanOrEqual	Determines whether one `OracleIntervalDS` value is less than or equal to another
NotEquals	Determines whether two `OracleIntervalDS` values are not equal
Parse	Returns an `OracleIntervalDS` structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an `OracleIntervalDS` with the specified day precision and fractional second precision

OracleIntervalDS Static Operators

The `OracleIntervalDS` static operators are listed in Table 5–51.

Table 5–51 OracleIntervalDS Static Operators

Operator	Description
operator +	Adds two `OracleIntervalDS` values
operator ==	Determines whether two `OracleIntervalDS` values are equal
operator >	Determines whether one `OracleIntervalDS` value is greater than another
operator >=	Determines whether one `OracleIntervalDS` value is greater than or equal to another
operator !=	Determines whether two `OracleIntervalDS` values are not equal
operator <	Determines whether one `OracleIntervalDS` value is less than another
operator <=	Determines whether one `OracleIntervalDS` value is less than or equal to another
operator -	Subtracts one `OracleIntervalDS` value from another
operator -	Negates an `OracleIntervalDS` structure
operator *	Multiplies an `OracleIntervalDS` value by a number
operator /	Divides an `OracleIntervalDS` value by a number

OracleIntervalDS Type Conversions

The `OracleIntervalDS` type conversions are listed in Table 5–52.

Table 5–52 OracleIntervalDS Type Conversions

Operator	Description
explicit operator TimeSpan	Converts an `OracleIntervalDS` structure to a `TimeSpan` structure
explicit operator OracleIntervalDS	Converts a string to an `OracleIntervalDS` structure
implicit operator OracleIntervalDS	Converts a `TimeSpan` structure to an `OracleIntervalDS` structure

OracleIntervalDS Properties

The `OracleIntervalDS` properties are listed in Table 5–53.

Table 5–53 OracleIntervalDS Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle `INTERVAL DAY TO SECOND` in Oracle internal format
Days	Gets the days component of an `OracleIntervalDS`
Hours	Gets the hours component of an `OracleIntervalDS`
IsNull	Indicates whether the current instance has a null value
Milliseconds	Gets the milliseconds component of an `OracleIntervalDS`
Minutes	Gets the minutes component of an `OracleIntervalDS`
Nanoseconds	Gets the nanoseconds component of an `OracleIntervalDS`
Seconds	Gets the seconds component of an `OracleIntervalDS`
TotalDays	Returns the total number, in days, that represent the time period in the `OracleIntervalDS` structure
Value	Specifies the time interval that is stored in the `OracleIntervalDS` structure

OracleIntervalDS Methods

The `OracleIntervalDS` methods are listed in Table 5–54.

Table 5–54 OracleIntervalDS Methods

Methods	Description
CompareTo	Compares the current `OracleIntervalDS` instance to an object, and returns an integer that represents their relative values

Table 5–54 OracleIntervalDS Methods (Cont.)

Methods	Description
Equals	Determines whether the specified `object` has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleIntervalDS` instance
GetType	Inherited from `Object`
ToString	Converts the current `OracleIntervalDS` structure to a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure

OracleIntervalDS Constructors

`OracleIntervalDS` constructors create a new instance of the `OracleIntervalDS` structure.

Overload List:

- OracleIntervalDS(TimeSpan)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a `TimeSpan` structure.

- OracleIntervalDS(string)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a string that indicates a period of time.

- OracleIntervalDS(double)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the total number of days.

- OracleIntervalDS(int, int, int, int, double)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds and milliseconds.

- OracleIntervalDS(int, int, int, int, int)

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds, and nanoseconds.

- OracleIntervalDS(byte[ ])

  This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value to the provided byte array, which is in an internal Oracle `INTERVAL DAY TO SECOND` format.

  See Also:

  - Oracle.DataAccess.Types Namespace (ODP.NET Types)
  - OracleIntervalDS Structure
  - OracleIntervalDS Members

OracleIntervalDS(TimeSpan)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a `TimeSpan` structure.

Declaration

```
// C#
public OracleIntervalDS(TimeSpan ts);
```

Parameters

- *ts*

  A `TimeSpan` structure.

  See Also:

  - Oracle.DataAccess.Types Namespace (ODP.NET Types)
  - OracleIntervalDS Structure
  - OracleIntervalDS Members

OracleIntervalDS(string)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using a string that indicates a period of time.

Declaration

```
// C#
public OracleIntervalDS(string intervalStr);
```

Parameters

- *intervalStr*

  A string representing the Oracle INTERVAL DAY TO SECOND.

Exceptions

ArgumentException - The *intervalStr* parameter is not in the valid format or has an invalid value.

ArgumentNullException - The *intervalStr* parameter is null.

Remarks

The value specified in the supplied *intervalStr* must be in Day HH:MI:SSxFF format.

Example

"1 2:3:4.99" means 1 day, 2 hours, 3 minutes, 4 seconds, and 990 milliseconds or 1 day, 2 hours, 3 minutes, 4 seconds, and 990000000 nanoseconds.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS(double)

This constructor creates a new instance of the OracleIntervalDS structure and sets its value using the total number of days.

Declaration

```
// C#
public OracleIntervalDS(double totalDays);
```

Parameters

- *totalDays*

The supplied total number of days for a time interval. Range of days is -1000,000,000 < *totalDays* < 1000,000,000.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleIntervalDS`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS(int, int, int, int, double)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds, and milliseconds.

Declaration

```
// C#
public OracleIntervalDS (int days, int hours, int minutes, int seconds,
 double milliSeconds);
```

Parameters

- *days*

  The days provided. Range of day is (-999,999,999 to 999,999,999).

- *hours*

  The hours provided. Range of hour is (-23 to 23).

- *minutes*

  The minutes provided. Range of minute is (-59 to 59).

- *seconds*

  The seconds provided. Range of second is (-59 to 59).

- *milliSeconds*

The milliseconds provided. Range of millisecond is (- 999.999999 to 999.999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleIntervalDS`.

Remarks

The sign of all the arguments must be the same.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS(int, int, int, int, int)

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value using the supplied days, hours, minutes, seconds, and nanoseconds.

Declaration

```
// C#
public OracleIntervalDS (int days, int hours, int minutes, int seconds,
 int nanoseconds);
```

Parameters

- *days*

  The days provided. Range of day is (-999,999,999 to 999,999,999).

- *hours*

  The hours provided. Range of hour is (-23 to 23).

- *minutes*

  The minutes provided. Range of minute is (-59 to 59).

- *seconds*

The seconds provided. Range of second is (-59 to 59).

- *nanoseconds*

  The nanoseconds provided. Range of nanosecond is (-999,999,999 to 999,999,999)

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleIntervalDS`.

Remarks

The sign of all the arguments must be the same.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS(byte[ ])

This constructor creates a new instance of the `OracleIntervalDS` structure and sets its value to the provided byte array, which is in an internal Oracle `INTERVAL DAY TO SECOND` format.

Declaration

```
// C#
public OracleIntervalDS (byte[ ] bytes);
```

Parameters

- *bytes*

  A byte array that is in an internal Oracle `INTERVAL DAY TO SECOND` format.

Exceptions

`ArgumentException` - `bytes` is not in internal Oracle `INTERVAL DAY TO SECOND` format, or `bytes` is not a valid Oracle `INTERVAL DAY TO SECOND`.

`ArgumentNullException` - *`bytes`* is null.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS Static Fields

The `OracleIntervalDS` static fields are listed in Table 5–55.

Table 5–55 OracleIntervalDS Static Fields

Field	Description
MaxValue	Represents the maximum valid time interval for an `OracleIntervalDS` structure
MinValue	Represents the minimum valid time interval for an `OracleIntervalDS` structure
Null	Represents a null value that can be assigned to an `OracleIntervalDS` instance
Zero	Represents a zero value for an `OracleIntervalDS` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

MaxValue

This static field represents the maximum value for an `OracleIntervalDS` structure.

Declaration

```
// C#
public static readonly OracleIntervalDS MaxValue;
```

Remarks

Maximum values:

- Day: 999999999
- hour: 23
- minute is 59
- second: 59
- nanosecond: 999999999

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

MinValue

This static field represents the minimum value for an `OracleIntervalDS` structure.

Declaration

```
// C#
public static readonly OracleIntervalDS MinValue;
```

Remarks

Minimum values:

- Day: -999999999
- hour: -23
- minute: -59
- second: -59
- nanosecond: -999999999

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Null

This static field represents a null value that can be assigned to an `OracleIntervalDS` instance.

Declaration

```
// C#
public static readonly OracleIntervalDS Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Zero

This static field represents a zero value for an `OracleIntervalDS` structure.

Declaration

```
// C#
public static readonly OracleIntervalDS Zero;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS Static Methods

The `OracleIntervalDS` static methods are listed in Table 5–56.

Table 5–56 OracleIntervalDS Static Methods

Methods	Description
Equals	Determines whether two `OracleIntervalDS` values are equal (Overloaded)
GreaterThan	Determines whether one `OracleIntervalDS` value is greater than another
GreaterThanOrEqual	Determines whether one `OracleIntervalDS` value is greater than or equal to another

Table 5–56 OracleIntervalDS Static Methods (Cont.)

Methods	Description
LessThan	Determines whether one `OracleIntervalDS` value is less than another
LessThanOrEqual	Determines whether one `OracleIntervalDS` value is less than or equal to another
NotEquals	Determines whether two `OracleIntervalDS` values are not equal
Parse	Returns an `OracleIntervalDS` structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an `OracleIntervalDS` with the specified day precision and fractional second precision

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Equals

This static method determines whether two `OracleIntervalDS` values are equal.

Declaration

```
// C#
public static bool Equals(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

If the two `OracleIntervalDS` structures represent the same time interval, returns `true`; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

GreaterThan

This static method determines whether the first of two `OracleIntervalDS` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

GreaterThanOrEqual

This static method determines whether the first of two `OracleIntervalDS` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleIntervalDS val1,
   OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

LessThan

This static method determines whether the first of two `OracleIntervalDS` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

LessThanOrEqual

This static method determines whether the first of two `OracleIntervalDS` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleIntervalDS val1, OracleIntervalDS
```

```
val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

NotEquals

This static method determines whether two `OracleIntervalDS` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second OracleIntervalDS.

Return Value

Returns true if two OracleIntervalDS values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalDS that has a value compares greater than an OracleIntervalDS that has a null value.
- Two OracleIntervalDSs that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Parse

This static method returns an OracleIntervalDS instance and sets its value for time interval using a string.

Declaration

```
// C#
public static OracleIntervalDS Parse(string intervalStr);
```

Parameters

- *intervalStr*

  A string representing the Oracle INTERVAL DAY TO SECOND.

Return Value

Returns an OracleIntervalDS instance representing the time interval from the supplied string.

Exceptions

`ArgumentException` - The *`intervalStr`* parameter is not in the valid format or *`intervalStr`* has an invalid value.

`ArgumentNullException` - The *`intervalStr`* parameter is null.

Remarks

The value specified in *`intervalStr`* must be in Day HH:MI:SSxFF format.

Example

"`1 2:3:4.99`" means 1 day, 2 hours, 3 minutes, 4 seconds, and 990 milliseconds or 1 day, 2 hours, 3 minutes, 4 seconds, and 990000000 nanoseconds.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

SetPrecision

This static method returns a new instance of an `OracleIntervalDS` with the specified day precision and fractional second precision.

Declaration

```
// C#
public static OracleIntervalDS SetPrecision(OracleIntervalDS value1,int
dayPrecision, int fracSecPrecision);
```

Parameters

- *`value1`*

  An `OracleIntervalDS` structure.

- *`dayPrecision`*

  The day precision provided. Range of day precision is (0 to 9).

- *`fracSecPrecision`*

  The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleIntervalDS` instance.

Exceptions

`ArgumentOutOfRangeException` - An argument value is out of the specified range.

Remarks

Depending on the value specified in the supplied *`dayPrecision`*, 0 or more leading zeros are displayed in the string returned by `ToString()`.

The value specified in the supplied *`fracSecPrecision`* is used to perform a rounding off operation on the supplied `OracleIntervalDS` value. Depending on this value, `0` or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleIntervalDS` with a value of "`1 2:3:4.99`" results in the string "`001 2:3:4.99000`" when `SetPrecision()` is called, with the day precision set to `3` and fractional second precision set to `5`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS Static Operators

The `OracleIntervalDS` static operators are listed in Table 5–57.

Table 5–57 OracleIntervalDS Static Operators

Operator	Description
operator +	Adds two `OracleIntervalDS` values
operator ==	Determines whether two `OracleIntervalDS` values are equal
operator >	Determines whether one `OracleIntervalDS` value is greater than another

Table 5–57 OracleIntervalDS Static Operators (Cont.)

Operator	Description
operator >=	Determines whether one `OracleIntervalDS` value is greater than or equal to another
operator !=	Determines whether two `OracleIntervalDS` values are not equal
operator <	Determines whether one `OracleIntervalDS` value is less than another
operator <=	Determines whether one `OracleIntervalDS` value is less than or equal to another
operator -	Subtracts one `OracleIntervalDS` value from another
operator -	Negates an `OracleIntervalDS` structure
operator *	Multiplies an `OracleIntervalDS` value by a number
operator /	Divides an `OracleIntervalDS` value by a number

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator +

This static operator adds two `OracleIntervalDS` values.

Declaration

```
// C#
public static OracleIntervalDS operator + (OracleIntervalDS val1,
OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

An `OracleIntervalDS`.

Remarks

If either argument has a null value, the returned `OracleIntervalDS` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator ==

This static operator determines if two `OracleIntervalDS` values are equal.

Declaration

```
// C#
public static bool operator == (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the two `OracleIntervalDS` values are the same; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator >

This static operator determines if the first of two `OracleIntervalDS` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if one `OracleIntervalDS` value is greater than another; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator >=

This static operator determines if the first of two `OracleIntervalDS` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator !=

This static operator determines if the two `OracleIntervalDS` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the two `OracleIntervalDS` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator <

This static operator determines if the first of two `OracleIntervalDS` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator <=

This static operator determines if the first of two `OracleIntervalDS` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

Returns `true` if the first of two `OracleIntervalDS` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator -

This static operator subtracts one `OracleIntervalDS` structure from another.

Declaration

```
// C#
public static OracleIntervalDS operator - (OracleIntervalDS val1,
OracleIntervalDS val2);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *val2*

  Second `OracleIntervalDS`.

Return Value

An `OracleIntervalDS` structure.

Remarks

If either argument has a null value, the returned `OracleIntervalDS` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator -

This static operator negates the supplied `OracleIntervalDS` structure.

Declaration

```
// C#
public static OracleIntervalDS operator - (OracleIntervalDS val);
```

Parameters

- *val*

  An `OracleIntervalDS`.

Return Value

An `OracleIntervalDS` structure.

Remarks

If the supplied `OracleIntervalDS` structure has a null value, the returned `OracleIntervalDS` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator *

This static operator multiplies an `OracleIntervalDS` value by a number.

Declaration

```
// C#
public static OracleIntervalDS operator * (OracleIntervalDS val1, int
multiplier);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *multiplier*

  A multiplier.

Return Value

A new `OracleIntervalDS` instance.

Remarks

If the `OracleIntervalDS` structure has a null value, the returned `OracleIntervalDS` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator /

This static operator divides an `OracleIntervalDS` value by a number.

Declaration

```
// C#
public static OracleIntervalDS operator / (OracleIntervalDS val1, int divisor);
```

Parameters

- *val1*

  First `OracleIntervalDS`.

- *divisor*

  A divisor.

Return Value

An `OracleIntervalDS` structure.

Remarks

If the `OracleIntervalDS` structure has a null value, the returned `OracleIntervalDS` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS Type Conversions

The `OracleIntervalDS` type conversions are listed in Table 5–58.

Table 5–58 OracleIntervalDS Type Conversions

Operator	Description
explicit operator TimeSpan	Converts an `OracleIntervalDS` structure to a `TimeSpan` structure
explicit operator OracleIntervalDS	Converts a string to an `OracleIntervalDS` structure
implicit operator OracleIntervalDS	Converts a `TimeSpan` structure to an `OracleIntervalDS` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

explicit operator TimeSpan

This type conversion operator converts an `OracleIntervalDS` structure to a `TimeSpan` structure.

Declaration

```
// C#
public static explicit operator TimeSpan(OracleIntervalDS val);
```

Parameters

- *val*

  An `OracleIntervalDS` instance.

Return Value

A `TimeSpan` structure.

Exceptions

`OracleNullValueException` - The `OracleIntervalDS` structure has a null value.

Remarks

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

explicit operator OracleIntervalDS

This type conversion operator converts a string to an `OracleIntervalDS` structure.

Declaration

```
// C#
public static explicit operator OracleIntervalDS (string intervalStr);
```

Parameters

- *`intervalStr`*

  A string representation of an Oracle `INTERVAL DAY TO SECOND`.

Return Value

An `OracleIntervalDS` structure.

Exceptions

`ArgumentException` - The supplied *`intervalStr`* parameter is not in the correct format or has an invalid value.

`ArgumentNullException` - The *`intervalStr`* parameter is null.

Remarks

The returned `OracleIntervalDS` structure contains the same time interval represented by the supplied *`intervalStr`*. The value specified in the supplied *`intervalStr`* must be in Day HH:MI:SSxFF format.

Example

`"1 2:3:4.99"` means 1 day, 2 hours, 3 minutes 4 seconds and 990 milliseconds or 1 day, 2 hours, 3 minutes 4 seconds and 990000000 nanoseconds.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

implicit operator OracleIntervalDS

This type conversion operator converts a `TimeSpan` structure to an `OracleIntervalDS` structure.

Declaration

```
// C#
public static implicit operator OracleIntervalDS(TimeSpan val);
```

Parameters

- *`val`*

  A `TimeSpan` instance.

Return Value

An `OracleIntervalDS` structure.

Remarks

The returned `OracleIntervalDS` structure contains the same days, hours, seconds, and milliseconds as the supplied `TimeSpan val`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS Properties

The `OracleIntervalDS` properties are listed in Table 5–59.

Table 5–59 OracleIntervalDS Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle `INTERVAL DAY TO SECOND` in Oracle internal format
Days	Gets the days component of an `OracleIntervalDS`
Hours	Gets the hours component of an `OracleIntervalDS`
IsNull	Indicates whether the current instance has a null value
Milliseconds	Gets the milliseconds component of an `OracleIntervalDS`
Minutes	Gets the minutes component of an `OracleIntervalDS`
Nanoseconds	Gets the nanoseconds component of an `OracleIntervalDS`
Seconds	Gets the seconds component of an `OracleIntervalDS`
TotalDays	Returns the total number, in days, that represent the time period in the `OracleIntervalDS` structure
Value	Specifies the time interval that is stored in the `OracleIntervalDS` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

BinData

This property returns an array of bytes that represents the Oracle `INTERVAL DAY TO SECOND` in Oracle internal format.

Declaration

```
// C#
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle `INTERVAL DAY TO SECOND` in Oracle internal format.

Exceptions

`OracleNullValueException` - The current instance has a null value.

Remarks

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Days

This property gets the days component of an `OracleIntervalDS`.

Declaration

```
// C#
public int Days {get;}
```

Property Value

An `int` representing the days component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Hours

This property gets the hours component of an `OracleIntervalDS`.

Declaration

```
// C#
public int Hours {get;}
```

Property Value

An `int` representing the hours component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

IsNull

This property indicates whether the current instance has a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

Returns `true` if the current instance has a null value; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Milliseconds

This property gets the milliseconds component of an `OracleIntervalDS`.

Declaration

```
// C#
public double Milliseconds {get;}
```

Property Value

A `double` that represents milliseconds component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Minutes

This property gets the minutes component of an `OracleIntervalDS`.

Declaration

```
// C#
public int Minutes {get;}
```

Property Value

A `int` that represents minutes component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Nanoseconds

This property gets the nanoseconds component of an `OracleIntervalDS`.

Declaration

```
// C#
public int Nanoseconds {get;}
```

Property Value

An `int` that represents nanoseconds component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Seconds

This property gets the seconds component of an `OracleIntervalDS`.

Declaration

```
// C#
public int Seconds {get;}
```

Property Value

An `int` that represents seconds component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

TotalDays

This property returns the total number, in days, that represent the time period in the `OracleIntervalDS` structure.

Declaration

```
// C#
public double TotalDays {get;}
```

Property Value

A `double` that represents the total number of days.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Value

This property specifies the time interval that is stored in the `OracleIntervalDS` structure.

Declaration

```
// C#
public TimeSpan Value {get;}
```

Property Value

A time interval.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalDS Methods

The `OracleIntervalDS` methods are listed in Table 5–60.

Table 5–60 OracleIntervalDS Methods

Methods	Description
CompareTo	Compares the current `OracleIntervalDS` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether the specified `object` has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleIntervalDS` instance
GetType	Inherited from `Object`
ToString	Converts the current `OracleIntervalDS` structure to a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

CompareTo

This method compares the current `OracleIntervalDS` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  The object being compared to.

Return Value

The method returns:

- Less than zero: if the current `OracleIntervalDS` represents a shorter time interval than *obj*.

- Zero: if the current `OracleIntervalDS` and *`obj`* represent the same time interval.
- Greater than zero: if the current `OracleIntervalDS` represents a longer time interval than *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* parameter is not of type `OracleIntervalDS`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleIntervalDS`s. For example, comparing an `OracleIntervalDS` instance with an `OracleBinary` instance is not allowed. When an `OracleIntervalDS` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

Equals

This method determines whether the specified `object` has the same time interval as the current instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *`obj`*

The specified object.

Return Value

Returns `true` if *`obj`* is of type `OracleIntervalDS` and has the same time interval as the current instance; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalDS` that has a value compares greater than an `OracleIntervalDS` that has a null value.
- Two `OracleIntervalDS`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleIntervalDS` instance.

Declaration

```
// C#
public override int GetHashCode();
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

ToString

Overrides `Object`

This method converts the current `OracleIntervalDS` structure to a string.

Declaration

```
// C#
public override string ToString();
```

Return Value

Returns a `string`.

Remarks

If the current instance has a null value, the returned string contains "null".

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalDS Structure
- OracleIntervalDS Members

OracleIntervalYM Structure

The `OracleIntervalYM` structure represents the Oracle `INTERVAL YEAR TO MONTH` datatype to be stored in or retrieved from a database. Each `OracleIntervalYM` stores a period of time in years and months.

Class Inheritance

```
Object
  ValueType
    OracleIntervalYM
```

Declaration

```
// C#
public struct OracleIntervalYM : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
//  Illustrates usage of OracleIntervalYM

OracleIntervalYM iymMax = OracleIntervalYM.MaxValue;
double maxYears = iymMax.TotalYears;
maxYears -= 1;
OracleIntervalYM iymMax_1 = new OracleIntervalYM(maxYears);

// Calculate the difference. It should be 1 +/- epsilon years
// where epsilon for OracleIntervalYM = 1 month.

OracleIntervalYM iymDiff = iymMax - iymMax_1;

// If the difference isnt exactly 1 day, display the difference
if (iymDiff.TotalYears != 1)
    Console.WriteLine(iymDiff.ToString());
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Members
- OracleIntervalYM Constructors
- OracleIntervalYM Static Fields
- OracleIntervalYM Static Methods
- OracleIntervalYM Static Operators
- OracleIntervalYM Type Conversions
- OracleIntervalYM Properties
- OracleIntervalYM Methods

OracleIntervalYM Members

`OracleIntervalYM` members are listed in the following tables:

OracleIntervalYM Constructors

`OracleIntervalYM` constructors are listed in Table 5–61

Table 5–61 OracleIntervalYM Constructors

Constructor	Description
OracleIntervalYM Constructors	Instantiates a new instance of `OracleIntervalYM` structure (Overloaded)

OracleIntervalYM Static Fields

The `OracleIntervalYM` static fields are listed in Table 5–62.

Table 5–62 OracleIntervalYM Static Fields

Field	Description
MaxValue	Represents the maximum value for an `OracleIntervalYM` structure

Table 5–62 OracleIntervalYM Static Fields (Cont.)

Field	Description
MinValue	Represents the minimum value for an `OracleIntervalYM` structure
Null	Represents a null value that can be assigned to an `OracleIntervalYM` instance
Zero	Represents a zero value for an `OracleIntervalYM` structure

OracleIntervalYM Static Methods

The `OracleIntervalYM` static methods are listed in Table 5–63.

Table 5–63 OracleIntervalYM Static Methods

Methods	Description
Equals	Determines whether two `OracleIntervalYM` values are equal (Overloaded)
GreaterThan	Determines whether one `OracleIntervalYM` value is greater than another
GreaterThanOrEqual	Determines whether one `OracleIntervalYM` value is greater than or equal to another
LessThan	Determines whether one `OracleIntervalYM` value is less than another
LessThanOrEqual	Determines whether one `OracleIntervalYM` value is less than or equal to another
NotEquals	Determines whether two `OracleIntervalYM` values are not equal
Parse	Returns an `OracleIntervalYM` structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an `OracleIntervalYM` with the specified year precision.

OracleIntervalYM Static Operators

The `OracleIntervalYM` static operators are listed in Table 5–64.

Table 5–64 OracleIntervalYM Static Operators

Operator	Description
operator +	Adds two `OracleIntervalYM` values
operator ==	Determines whether two `OracleIntervalYM` values are equal
operator >	Determines whether one `OracleIntervalYM` value is greater than another
operator >=	Determines whether one `OracleIntervalYM` value is greater than or equal to another
operator !=	Determines whether two `OracleIntervalYM` values are not equal
operator <	Determines whether one `OracleIntervalYM` value is less than another
operator <=	Determines whether one `OracleIntervalYM` value is less than or equal to another
operator -	Subtracts one `OracleIntervalYM` value from another
operator -	Negates an `OracleIntervalYM` structure
operator *	Multiplies an `OracleIntervalYM` value by a number
operator /	Divides an `OracleIntervalYM` value by a number

OracleIntervalYM Type Conversions

The `OracleIntervalYM` conversions are listed in Table 5–65.

Table 5–65 OracleIntervalYM Type Conversions

Operator	Description
explicit operator long	Converts an `OracleIntervalYM` structure to a number
explicit operator OracleIntervalYM	Converts a string to an `OracleIntervalYM` structure
implicit operator OracleIntervalYM	Converts the number of months to an `OracleIntervalYM` structure

OracleIntervalYM Properties

The `OracleIntervalYM` properties are listed in Table 5–66.

Table 5–66 OracleIntervalYM Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle `INTERVAL YEAR TO MONTH` in an Oracle internal format
IsNull	Indicates whether the current instance has a null value
Months	Gets the months component of an `OracleIntervalYM`
TotalYears	Returns the total number, in years, that represents the period of time in the current `OracleIntervalYM` structure
Value	Specifies the total number of months that is stored in the `OracleIntervalYM` structure
Years	Gets the years component of an `OracleIntervalYM`

OracleIntervalYM Methods

The `OracleIntervalYM` methods are listed in Table 5–67.

Table 5–67 OracleIntervalYM Methods

Methods	Description
CompareTo	Compares the current `OracleIntervalYM` instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether the specified `object` has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleIntervalYM` instance
GetType	Inherited from `Object`
ToString	Converts the current `OracleIntervalYM` structure to a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure

OracleIntervalYM Constructors

The `OracleIntervalYM` constructors creates a new instance of the `OracleIntervalYM` structure.

Overload List:

- OracleIntervalYM(long)

  This method creates a new instance of the `OracleIntervalYM` structure using the supplied total number of months for a period of time.

- OracleIntervalYM(string)

  This method creates a new instance of the `OracleIntervalYM` structure and sets its value using the supplied string.

- OracleIntervalYM(double)

  This method creates a new instance of the `OracleIntervalYM` structure and sets its value using the total number of years.

- OracleIntervalYM(int, int)

  This method creates a new instance of the `OracleIntervalYM` structure and sets its value using years and months.

- OracleIntervalYM(byte[ ])

  This method creates a new instance of the `OracleIntervalYM` structure and sets its value to the provided byte array, which is in an internal Oracle `INTERVAL DAY TO SECOND` format.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM(long)

This method creates a new instance of the `OracleIntervalYM` structure using the supplied total number of months for a period of time.

Declaration

```
// C#
public OracleIntervalYM (long totalMonths);
```

Parameters

- *totalMonths*

  The number of total months for a time interval. Range is -12,000,000,000 < *totalMonths* < 12,000,000,000.

Exceptions

`ArgumentOutOfRangeException` - The *totalMonths* parameter is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM(string)

This method creates a new instance of the `OracleIntervalYM` structure and sets its value using the supplied string.

Declaration

```
// C#
public OracleIntervalYM (string intervalStr);
```

Parameters

- *intervalStr*

  A string representing the Oracle `INTERVAL YEAR TO MONTH`.

Remarks

The value specified in the supplied *intervalStr* must be in Year-Month format.

Exceptions

`ArgumentException` - The *`intervalStr`* parameter is not in the valid format or *`intervalStr`* has an invalid value.

`ArgumentNullException` - The *`intervalStr`* parameter is null.

Example

"1-2" means 1 year and 2 months.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM(double)

This method creates a new instance of the `OracleIntervalYM` structure and sets its value using the total number of years.

Declaration

```
// C#
public OracleIntervalYM (double totalYears);
```

Parameters

- *`totalYears`*

  Number of total years. Range is -1,000,000,000 < *`totalYears`* > 1,000,000,000.

Exceptions

`ArgumentOutOfRangeException` - The *`totalYears`* parameter is out of the specified range.

`ArgumentException` - The *`totalYears`* parameter cannot be used to construct a valid `OracleIntervalYM`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM(int, int)

This method creates a new instance of the `OracleIntervalYM` structure and sets its value using years and months.

Declaration

```
// C#
public OracleIntervalYM (int years, int months);
```

Parameters

- *years*

  Number of years. Range of year is (-999,999,999 to 999,999,999).

- *months*

  Number of months. Range of month is (-11 to 11).

Remarks

The sign of all the arguments must be the same.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleIntervalYM`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM(byte[ ])

This method creates a new instance of the `OracleIntervalYM` structure and sets its value to the provided byte array, which is in an internal Oracle `INTERVAL DAY TO SECOND` format.

Declaration

```
// C#
public OracleIntervalYM (byte[] bytes);
```

Parameters

- *bytes*

  A byte array that is in an internal Oracle INTERVAL YEAR TO MONTH format.

Exceptions

ArgumentException - The supplied byte array is not in an internal Oracle INTERVAL YEAR TO MONTH format or the supplied byte array has an invalid value.

ArgumentNullException - *bytes* is null.

Remarks

The supplied byte array must be in an internal Oracle INTERVAL YEAR TO MONTH format.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM Static Fields

The OracleIntervalYM static fields are listed in Table 5–68.

Table 5–68 OracleIntervalYM Static Fields

Field	Description
MaxValue	Represents the maximum value for an OracleIntervalYM structure
MinValue	Represents the minimum value for an OracleIntervalYM structure
Null	Represents a null value that can be assigned to an OracleIntervalYM instance
Zero	Represents a zero value for an OracleIntervalYM structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

MaxValue

This static field represents the maximum value for an `OracleIntervalYM` structure.

Declaration

```
// C#
public static readonly OracleIntervalYM MaxValue;
```

Remarks

Year is 999999999 and Month is 11.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

MinValue

This static field represents the minimum value for an `OracleIntervalYM` structure.

Declaration

```
// C#
public static readonly OracleIntervalYM MinValue;
```

Remarks

Year is -999999999 and Month is -11.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

Null

This static field represents a null value that can be assigned to an `OracleIntervalYM` instance.

Declaration

```
// C#
public static readonly OracleIntervalYM Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

Zero

This static field represents a zero value for an `OracleIntervalYM` structure.

Declaration

```
// C#
public static readonly OracleIntervalDS Zero;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM Static Methods

The `OracleIntervalYM` static methods are listed in Table 5–69.

Table 5–69 OracleIntervalYM Static Methods

Methods	Description
Equals	Determines whether two `OracleIntervalYM` values are equal (Overloaded)
GreaterThan	Determines whether one `OracleIntervalYM` value is greater than another
GreaterThanOrEqual	Determines whether one `OracleIntervalYM` value is greater than or equal to another
LessThan	Determines whether one `OracleIntervalYM` value is less than another
LessThanOrEqual	Determines whether one `OracleIntervalYM` value is less than or equal to another

Table 5–69 OracleIntervalYM Static Methods (Cont.)

Methods	Description
NotEquals	Determines whether two `OracleIntervalYM` values are not equal
Parse	Returns an `OracleIntervalYM` structure and sets its value for time interval using a string
SetPrecision	Returns a new instance of an `OracleIntervalYM` with the specified year precision.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

Equals

This static method determines whether two `OracleIntervalYM` values are equal.

Declaration

```
// C#
public static bool Equals(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  An `OracleIntervalYM` structure.

- *val2*

  An `OracleIntervalYM` structure.

Return Value

Returns `true` if two `OracleIntervalYM` values represent the same time interval, otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

GreaterThan

This static method determines whether the first of two `OracleIntervalYM` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

GreaterThanOrEqual

This static method determines whether the first of two `OracleIntervalYM` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleIntervalYM val1, OracleIntervalYM
val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is greater than or equal to the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

LessThan

This static method determines whether the first of two `OracleIntervalYM` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

LessThanOrEqual

This static method determines whether the first of two `OracleIntervalYM` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleIntervalYM val1, OracleIntervalYM
```

```
val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is less than or equal to the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

NotEquals

This static method determines whether two `OracleIntervalYM` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second OracleIntervalYM.

Return Value

Returns true if two OracleIntervalYM values are not equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

Parse

This static method returns an OracleIntervalYM structure and sets its value for time interval using a string.

Declaration

```
// C#
public static OracleIntervalYM Parse (string intervalStr);
```

Parameters

- *intervalStr*

  A string representing the Oracle INTERVAL YEAR TO MONTH.

Return Value

Returns an OracleIntervalYM structure.

Exceptions

ArgumentException - The *intervalStr* parameter is not in the valid format or *intervalStr* has an invalid value.

`ArgumentNullException` - The *`intervalStr`* parameter is null.

Remarks

The value specified in the supplied *`intervalStr`* must be in the Year-Month format.

Example

"1-2" means 1 year and 2 months.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

SetPrecision

This static method returns a new instance of an `OracleIntervalYM` with the specified year precision.

Declaration

```
// C#
public static OracleIntervalYM SetPrecision(OracleIntervalYM value1,int
yearPrecision);
```

Parameters

- *`value1`*

  An `OracleIntervalYM` structure.

- *`yearPrecision`*

  The year precision provided. Range of year precision is (0 to 9).

Return Value

An `OracleIntervalDS` instance.

Exceptions

`ArgumentOutOfRangeException` - *`yearPrecision`* is out of the specified range.

Remarks

Depending on the value specified in the supplied `yearPrecision`, `0` or more leading zeros are displayed in the string returned by `ToString()`.

Example

An `OracleIntervalYM` with a value of "`1-2`" results in the string "`001-2`" when `SetPrecision()` is called with the year precision set to `3`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM Static Operators

The `OracleIntervalYM` static operators are listed in Table 5–70.

Table 5–70 OracleIntervalYM Static Operators

Operator	Description
operator +	Adds two `OracleIntervalYM` values
operator ==	Determines whether two `OracleIntervalYM` values are equal
operator >	Determines whether one `OracleIntervalYM` value is greater than another
operator >=	Determines whether one `OracleIntervalYM` value is greater than or equal to another
operator !=	Determines whether two `OracleIntervalYM` values are not equal
operator <	Determines whether one `OracleIntervalYM` value is less than another
operator <=	Determines whether one `OracleIntervalYM` value is less than or equal to another
operator -	Subtracts one `OracleIntervalYM` value from another
operator -	Negates an `OracleIntervalYM` structure
operator *	Multiplies an `OracleIntervalYM` value by a number
operator /	Divides an `OracleIntervalYM` value by a number

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator +

This static operator adds two `OracleIntervalYM` values.

Declaration

```
// C#
public static OracleIntervalYM operator + (OracleIntervalYM val1,
OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

`OracleIntervalYM`

Remarks

If either argument has a null value, the returned `OracleIntervalYM` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator ==

This static operator determines if two `OracleIntervalYM` values are equal.

Declaration

```
// C#
public static bool operator == (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

Returns `true` if they are equal; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator >

This static operator determines if the first of two `OracleIntervalYM` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second OracleIntervalYM.

Return Value

Returns true if one OracleIntervalYM value is greater than another; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleIntervalYM that has a value compares greater than an OracleIntervalYM that has a null value.
- Two OracleIntervalYMs that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator >=

This static operator determines if the first of two OracleIntervalYM values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  First OracleIntervalYM.

- *val2*

  Second OracleIntervalYM.

Return Value

Returns true if one OracleIntervalYM value is greater than or equal to another; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator !=

This static operator determines whether two `OracleIntervalYM` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleIntervalYM val1, OracleIntervalYM val2)
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

Returns `true` if two `OracleIntervalYM` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator <

This static operator determines if the first of two `OracleIntervalYM` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator <=

This static operator determines if the first of two `OracleIntervalYM` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleIntervalYM val1, OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

Returns `true` if the first of two `OracleIntervalYM` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator -

This static operator subtracts one `OracleIntervalYM` structure from another.

Declaration

```
// C#
public static OracleIntervalYM operator - (OracleIntervalYM val1,
OracleIntervalYM val2);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *val2*

  Second `OracleIntervalYM`.

Return Value

An `OracleIntervalYM` structure.

Remarks

If either argument has a null value, the returned `OracleIntervalYM` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator -

This static operator negates an `OracleIntervalYM` structure.

Declaration

```
// C#
public static OracleIntervalYM operator - (OracleIntervalYM val);
```

Parameters

- *val*

  An `OracleIntervalYM`.

Return Value

An `OracleIntervalYM` structure.

Remarks

If the supplied `OracleIntervalYM` structure has a null value, the returned `OracleIntervalYM` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator *

This static operator multiplies an `OracleIntervalYM` value by a number.

Declaration

```
// C#
public static OracleIntervalYM operator * (OracleIntervalYM val1, int
multiplier);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *multiplier*

  A multiplier.

Return Value

An `OracleIntervalYM` structure.

Remarks

If the supplied `OracleIntervalYM` structure has a null value, the returned `OracleIntervalYM` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator /

This static operator divides an `OracleIntervalYM` value by a number.

Declaration

```
// C#
public static OracleIntervalYM operator / (OracleIntervalYM val1, int divisor);
```

Parameters

- *val1*

  First `OracleIntervalYM`.

- *divisor*

  A divisor.

Return Value

An `OracleIntervalYM` structure.

Remarks

If the supplied `OracleIntervalYM` structure has a null value, the returned `OracleIntervalYM` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM Type Conversions

The `OracleIntervalYM` conversions are listed in Table 5–71.

Table 5–71 OracleIntervalYM Type Conversions

Operator	Description
explicit operator long	Converts an `OracleIntervalYM` structure to a number
explicit operator OracleIntervalYM	Converts a string to an `OracleIntervalYM` structure
implicit operator OracleIntervalYM	Converts the number of months to an `OracleIntervalYM` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

explicit operator long

This type conversion operator converts an `OracleIntervalYM` to a number that represents the number of months in the time interval.

Declaration

```
// C#
public static explicit operator long (OracleIntervalYM val);
```

Parameters

- *val*

  An `OracleIntervalYM` structure.

Return Value

A `long` number in months.

Exceptions

`OracleNullValueException` - The `OracleIntervalYM` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

explicit operator OracleIntervalYM

This type conversion operator converts the string *intervalStr* to an `OracleIntervalYM` structure.

Declaration

```
// C#
public static explicit operator OracleIntervalYM (string intervalStr);
```

Parameters

- *intervalStr*

  A string representation of an Oracle INTERVAL YEAR TO MONTH.

Return Value

An OracleIntervalYM structure.

Exceptions

ArgumentException - The supplied *intervalStr* parameter is not in the correct format or has an invalid value.

ArgumentNullException - The *intervalStr* parameter is null.

Remarks

The returned OracleIntervalDS structure contains the same time interval represented by the supplied *intervalStr*. The value specified in the supplied *intervalStr* must be in Year-Month format.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

implicit operator OracleIntervalYM

This type conversion operator converts the total number of months as time interval to an OracleIntervalYM structure.

Declaration

```
// C#
public static implicit operator OracleIntervalYM (long months);
```

Parameters

- *months*

  The number of months to be converted. Range is (-999,999,999 * 12)-11 <= *months* <= (999,999,999 * 12)+11.

Return Value

An `OracleIntervalYM` structure.

Exceptions

`ArgumentOutOfRangeException` - The *`months`* parameter is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM Properties

The `OracleIntervalYM` properties are listed in Table 5–72.

Table 5–72 OracleIntervalYM Properties

Properties	Description
BinData	Returns an array of bytes that represents the Oracle `INTERVAL YEAR TO MONTH` in an Oracle internal format
IsNull	Indicates whether the current instance has a null value
Months	Gets the months component of an `OracleIntervalYM`
TotalYears	Returns the total number, in years, that represents the period of time in the current `OracleIntervalYM` structure
Value	Specifies the total number of months that is stored in the `OracleIntervalYM` structure
Years	Gets the years component of an `OracleIntervalYM`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

BinData

This property returns an array of bytes that represents the Oracle `INTERVAL YEAR TO MONTH` in Oracle internal format.

Declaration

```
// C#
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle `INTERVAL YEAR TO MONTH` in Oracle internal format.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

IsNull

This property indicates whether the value has a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

Returns `true` if value has a null value; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

Months

This property gets the months component of an `OracleIntervalYM`.

Declaration

```
// C#
public int Months {get;}
```

Property Value

An `int` representing the months component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

TotalYears

This property returns the total number, in years, that represents the period of time in the current `OracleIntervalYM` structure.

Declaration

```
// C#
public double TotalYears {get;}
```

Property Value

A `double` representing the total number of years.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

Value

This property gets the total number of months that is stored in the `OracleIntervalYM` structure.

Declaration

```
// C#
public long Value {get;}
```

Property Value

The total number of months representing the time interval.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

Years

This property gets the years component of an `OracleIntervalYM`.

Declaration

```
// C#
public int Years {get;}
```

Property Value

An `int` representing the years component.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleIntervalYM Methods

The `OracleIntervalYM` methods are listed in Table 5–73.

Table 5–73 OracleIntervalYM Methods

Methods	Description
CompareTo	Compares the current `OracleIntervalYM` instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether the specified `object` has the same time interval as the current instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleIntervalYM` instance
GetType	Inherited from `Object`
ToString	Converts the current `OracleIntervalYM` structure to a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

CompareTo

This method compares the current `OracleIntervalYM` instance to the supplied object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  The supplied object.

Return Value

The method returns a number:

Less than zero: if the current `OracleIntervalYM` represents a shorter time interval than *obj*.

Zero: if the current `OracleIntervalYM` and *obj* represent the same time interval.

Greater than zero: if the current `OracleIntervalYM` represents a longer time interval than *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *obj* parameter is not of type `OracleIntervalYM`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleIntervalYM`s. For example, comparing an `OracleIntervalYM` instance with an `OracleBinary` instance is not allowed. When an `OracleIntervalYM` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

Equals

Overrides `Object`

This method determines whether the specified object has the same time interval as the current instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *obj*

  The supplied object.

Return Value

Returns `true` if the specified object instance is of type `OracleIntervalYM` and has the same time interval; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleIntervalYM` that has a value compares greater than an `OracleIntervalYM` that has a null value.
- Two `OracleIntervalYM`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleIntervalYM` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

An `int` representing a hash code.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

ToString

Overrides `Object`

This method converts the current `OracleIntervalYM` structure to a string.

Declaration

```
// C#
public override string ToString();
```

Return Value

A string that represents the current `OracleIntervalYM` structure.

Remarks

If the current instance has a null value, the returned string contain "null".

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleIntervalYM Structure
- OracleIntervalYM Members

OracleString Structure

The `OracleString` structure represents a variable-length stream of characters to be stored in or retrieved from a database.

Class Inheritance

```
Object
  ValueType
    OracleString
```

Declaration

```
// C#
public struct OracleString : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
// Illustrates the usage of OracleString

// bytes1 is non-Unicode encoded byte array = AAAAA
// bytes2 is Unicode encoded byte array = aa

byte[] bytes1 = new byte[] {65,65,65,65,65};
byte[] bytes2 = new byte[] {97,0,97,0};

// set str1 = AAA
// set str2 = a
OracleString str1 = new OracleString(bytes1, 0, 3, false, true);
OracleString str2 = new OracleString(bytes2, 2, 2, true, true);

// Display the constructed strings
Console.WriteLine("String str1 = " + str1.Value  +
 ". Length = " + str1.Length); // Prints String str1 = AAA. Length = 3

 Console.WriteLine("String str2 = " + str2.Value  +
   ". Length = " + str2.Length); // Prints String str2 = a. Length = 1
```

```
while (str1 > str2)
   str2 = OracleString.Concat(str2,"a");

// Display the constructed strings
Console.WriteLine("String str1 = " + str1.Value +
   ". Length = " + str1.Length); // Prints String str1 = AAA. Length= 3
Console.WriteLine("String str2 = " + str2.Value +
   ". Length = " + str2.Length); // Prints String str2 = aaa. Length= 3
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Members
- OracleString Constructors
- OracleString Static Fields
- OracleString Static Methods
- OracleString Static Operators
- OracleString Type Conversions
- OracleString Properties
- OracleString Methods

OracleString Members

`OracleString` members are listed in the following tables:

OracleString Constructors

`OracleString` constructors are listed in Table 5–74

Table 5–74 OracleString Constructors

Constructor	Description
OracleString Constructors	Instantiates a new instance of `OracleString` structure (Overloaded)

OracleString Static Fields

The `OracleString` static fields are listed in Table 5–75.

Table 5–75 OracleString Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the `OracleString` structure

OracleString Static Methods

The `OracleString` static methods are listed in Table 5–76.

Table 5–76 OracleString Static Methods

Methods	Description
Concat	Concatenates two `OracleString` instances and returns a new `OracleString` instance that represents the result
Equals	Determines if two `OracleString` values are equal (Overloaded)
GreaterThan	Determines whether the first of two `OracleString` values is greater than the second
GreaterThanOrEqual	Determines whether the first of two `OracleString` values is greater than or equal to the second
LessThan	Determines whether the first of two `OracleString` values is less than the second
LessThanOrEqual	Determines whether the first of two `OracleString` values is less than or equal to the second
NotEquals	Determines whether two `OracleString` values are not equal

OracleString Static Operators

The `OracleString` static operators are listed in Table 5–77.

Table 5–77 OracleString Static Operators

Operator	Description
operator +	Concatenates two `OracleString` values
operator ==	Determines if two `OracleString` values are equal
operator >	Determines if the first of two `OracleString` values is greater than the second
operator >=	Determines if the first of two `OracleString` values is greater than or equal to the second
operator !=	Determines if the two `OracleString` values are not equal
operator <	Determines if the first of two `OracleString` values is less than the second
operator <=	Determines if two `OracleString` values are not equal

OracleString Type Conversions

The `OracleString` type conversions are listed in Table 5–78.

Table 5–78 OracleString Type Conversions

Operator	Description
explicit operator string	Converts the supplied `OracleString` to a `string` instance
implicit operator OracleString	Converts the supplied `string` to an `OracleString` instance

OracleString Properties

The `OracleString` properties are listed in Table 5–79.

Table 5–79 OracleString Properties

Properties	Description
IsCaseIgnored	Indicates whether case should be ignored when performing string comparison

Table 5–79 OracleString Properties (Cont.)

Properties	Description
IsNull	Indicates whether the current instance has a null value
Item	Obtains the particular character in an `OracleString` using an index.
Length	Returns the length of the `OracleString`

OracleString Methods

The `OracleString` methods are listed in Table 5–80.

Table 5–80 OracleString Methods

Methods	Description
Clone	Returns a copy of the current `OracleString` instance
CompareTo	Compares the current `OracleString` instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether an object has the same string value as the current `OracleString` structure (Overloaded)
GetHashCode	Returns a hash code for the `OracleString` instance
GetNonUnicodeBytes	Returns an array of bytes, containing the contents of the `OracleString`, in the client character set format
GetType	Inherited from `Object`
GetUnicodeBytes	Returns an array of bytes, containing the contents of the `OracleString`, in Unicode format
ToString	Converts the current `OracleString` instance to a `string`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure

OracleString Constructors

The `OracleString` constructors create new instances of the `OracleString` structure.

Overload List:

- OracleString(string)

  This constructor creates a new instance of the `OracleString` structure and sets its value using a string.

- OracleString(string, bool)

  This constructor creates a new instance of the `OracleString` structure and sets its value using a string and specifies if case is ignored in comparison.

- OracleString(byte [ ], bool)

  This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array and specifies if the supplied byte array is Unicode encoded.

- OracleString(byte [ ], bool, bool)

  This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array and specifies the following: if the supplied byte array is Unicode encoded and if case is ignored in comparison.

- OracleString(byte [ ], int, int, bool)

  This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, and if the supplied byte array is Unicode encoded.

- OracleString(byte [ ], int, int, bool, bool)

  This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, if the supplied byte array is Unicode encoded, and if case is ignored in comparison.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString(string)

This constructor creates a new instance of the `OracleString` structure and sets its value using a string.

Declaration

```
// C#
public OracleString(string data);
```

Parameters

- *data*

  A string value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString(string, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a string and specifies if case is ignored in comparison.

Declaration

```
// C#
public OracleString(string data, bool isCaseIgnored);
```

Parameters

- *data*

  A string value.

- *isCaseIgnored*

Specifies if case is ignored in comparison. Specifies `true` if case is to be ignored; otherwise, specifies `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString(byte [ ], bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array and specifies if the supplied byte array is Unicode encoded.

Declaration

```
// C#
public OracleString(byte[] data, bool fUnicode);
```

Parameters

- *`data`*

  Byte array data for the new `OracleString`.

- *`fUnicode`*

  Specifies if the supplied `data` is Unicode encoded. Specifies `true` if Unicode encoded; otherwise, `false`.

Exceptions

`ArgumentNullException` - The *`data`* parameter is null.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString(byte [ ], bool, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array and specifies the following: if the supplied byte array is Unicode encoded and if case is ignored in comparison.

Declaration

```
// C#
public OracleString(byte[] data, bool fUnicode, bool isCaseIgnored);
```

Parameters

- *`data`*

  Byte array data for the new `OracleString`.

- *`fUnicode`*

  Specifies if the supplied `data` is Unicode encoded. Specifies `true` if Unicode encoded; otherwise, `false`.

- *`isCaseIgnored`*

  Specifies if case is ignored in comparison. Specifies `true` if case is to be ignored; otherwise, specifies `false`.

Exceptions

`ArgumentNullException` - The *`data`* parameter is null.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString(byte [ ], int, int, bool)

This constructor creates a new instance of the `OracleString` structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, and if the supplied byte array is Unicode encoded.

Declaration

```
// C#
public OracleString(byte[] data, int index, int count, bool fUnicode);
```

Parameters

- *data*

  Byte array data for the new OracleString.

- *index*

  The starting index to copy from *data*.

- *count*

  The number of bytes to copy.

- *fUnicode*

  Specifies if the supplied data is Unicode encoded. Specifies true if Unicode encoded; otherwise, false.

Exceptions

ArgumentNullException - The *data* parameter is null.

ArgumentOutOfRangeException - The *count* parameter is less than zero.

IndexOutOfRangeException - The *index* parameter is greater than or equal to the length of *data* or less than zero.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString(byte [ ], int, int, bool, bool)

This constructor creates a new instance of the OracleString structure and sets its value using a byte array, and specifies the following: the starting index in the byte array, the number of bytes to copy from the byte array, if the supplied byte array is Unicode encoded, and if case is ignored in comparison.

Declaration

```
// C#
public OracleString(byte[] data, int index, int count, bool fUnicode,
  bool isCaseIgnored);
```

Parameters

- *data*

  Byte array data for the new OracleString.

- *index*

  The starting index to copy from *data*.

- *count*

  The number of bytes to copy.

- *fUnicode*

  Specifies if the supplied data is Unicode encoded. Specifies true if Unicode encoded; otherwise, false.

- *isCaseIgnored*

  Specifies if case is ignored in comparison. Specifies true if case is to be ignored; otherwise, specifies false.

Exceptions

ArgumentNullException - The *data* parameter is null.

ArgumentOutOfRangeException - The *count* parameter is less than zero.

IndexOutOfRangeException - The *index* parameter is greater than or equal to the length of *data* or less than zero.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString Static Fields

The OracleString static fields are listed in Table 5–81.

Table 5–81 OracleString Static Fields

Field	Description
Null	Represents a null value that can be assigned to an instance of the OracleString structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

Null

This static field represents a null value that can be assigned to an instance of the `OracleString` structure.

Declaration

```
// C#
public static readonly OracleString Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString Static Methods

The `OracleString` static methods are listed in Table 5–82.

Table 5–82 OracleString Static Methods

Methods	Description
Concat	Concatenates two `OracleString` instances and returns a new `OracleString` instance that represents the result
Equals	Determines if two `OracleString` values are equal (Overloaded)
GreaterThan	Determines whether the first of two `OracleString` values is greater than the second
GreaterThanOrEqual	Determines whether the first of two `OracleString` values is greater than or equal to the second
LessThan	Determines whether the first of two `OracleString` values is less than the second
LessThanOrEqual	Determines whether the first of two `OracleString` values is less than or equal to the second

Table 5–82 OracleString Static Methods (Cont.)

Methods	Description
NotEquals	Determines whether two `OracleString` values are not equal

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

Concat

This static method concatenates two `OracleString` instances and returns a new `OracleString` instance that represents the result.

Declaration

```
// C#
public static OracleString Concat(OracleString str1, OracleString str2);
```

Parameters

- *str1*

  First `OracleString`.

- *str2*

  Second `OracleString`.

Return Value

An `OracleString`.

Remarks

If either argument has a null value, the returned `OracleString` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

Equals

Overloads `Object`

This static method determines whether the two `OracleStrings` being compared are equal.

Declaration

```
// C#
public static bool Equals(OracleString str1, OracleString str2);
```

Parameters

- *str1*

  First `OracleString`.

- *str2*

  Second `OracleString`.

Return Value

Returns `true` if the two `OracleStrings` being compared are equal; returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

GreaterThan

This static method determines whether the first of two `OracleString` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleString str1, OracleString str2);
```

Parameters

- *str1*

  First `OracleString`.

- *str2*

  Second `OracleString`.

Return Value

Returns `true` if the first of two `OracleStrings` is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

GreaterThanOrEqual

This static method determines whether the first of two `OracleString` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleString str1, OracleString str2);
```

Parameters

- *str1*

First `OracleString`.

- *`str2`*

  Second `OracleString`.

Return Value

Returns `true` if the first of two `OracleStrings` is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

LessThan

This static method determines whether the first of two `OracleString` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleString str1, OracleString str2);
```

Parameters

- *`str1`*

  First `OracleString`.

- *`str2`*

  Second `OracleString`.

Return Value

Returns `true` if the first is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

LessThanOrEqual

This static method determines whether the first of two `OracleString` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleString str1, OracleString str2);
```

Parameters

- *str1*

  First `OracleString`.

- *str2*

  Second `OracleString`.

Return Value

Returns `true` if the first is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any OracleString that has a value is greater than an OracleString that has a null value.
- Two OracleStrings that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

NotEquals

This static method determines whether two OracleString values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleString str1, OracleString str2);
```

Parameters

- *str1*

  First OracleString.

- *str2*

  Second OracleString.

Return Value

Returns true if the two OracleString instances are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleString that has a value is greater than an OracleString that has a null value.
- Two OracleStrings that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString Static Operators

The `OracleString` static operators are listed in Table 5–83.

Table 5–83 OracleString Static Operators

Operator	Description
operator +	Concatenates two `OracleString` values
operator ==	Determines if two `OracleString` values are equal
operator >	Determines if the first of two `OracleString` values is greater than the second
operator >=	Determines if the first of two `OracleString` values is greater than or equal to the second
operator !=	Determines if the two `OracleString` values are not equal
operator <	Determines if the first of two `OracleString` values is less than the second
operator <=	Determines if two `OracleString` values are not equal

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

operator +

This static operator concatenates two `OracleString` values.

Declaration

```
// C#
public static OracleString operator + (OracleString value1,  OracleString
value2);
```

Parameters

- *value1*

  First `OracleString`.

- *value2*

  Second `OracleString`.

Return Value

An `OracleString`.

Remarks

If either argument has a null value, the returned `OracleString` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

operator ==

This static operator determines if two `OracleString` values are equal.

Declaration

```
// C#
public static bool operator == (OracleString value1, OracleString value2);
```

Parameters

- *value1*

  First `OracleString`.

- *value2*

  Second `OracleString`.

Return Value

Returns `true` if two `OracleString` values are equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

operator >

This static operator determines if the first of two `OracleString` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleString value1, OracleString value2);
```

Parameters

- *`value1`*

  First `OracleString`.

- *`value2`*

  Second `OracleString`.

Return Value

Returns `true` if the first of two `OracleString` values is greater than the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

operator >=

This static operator determines if the first of two `OracleString` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleString value1, OracleString value2);
```

Parameters

- *value1*

  First `OracleString`.

- *value2*

  Second `OracleString`.

Return Value

Returns `true` if the first of two `OracleString` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

operator !=

This static operator determines if two `OracleString` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleString value1, OracleString value2);
```

Parameters

- *value1*

  First `OracleString`.

- *value2*

  Second `OracleString`.

Return Value

Returns `true` if two `OracleString` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

operator <

This static operator determines if the first of two `OracleStrings` is less than the second.

Declaration

```
// C#
public static bool operator < (OracleString value1, OracleString value2);
```

Parameters

- *value1*

  First `OracleString`.

- *value2*

  Second `OracleString`.

Return Value

Returns `true` if the first of two `OracleStrings` is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` has a null value.
- Two `OracleStrings` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

operator <=

This static operator determines if the first of two `OracleString` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleString value1, OracleString value1);
```

Parameters

- *value1*

  First `OracleString`.

- *value2*

  Second `OracleString`.

Return Value

Returns `true` if the first of two `OracleString` values is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString Type Conversions

The `OracleString` type conversions are listed in Table 5–84.

Table 5–84 OracleString Type Conversions

Operator	Description
explicit operator string	Converts the supplied `OracleString` to a `string` instance
implicit operator OracleString	Converts the supplied `string` to an `OracleString` instance

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

explicit operator string

This type conversion operator converts the supplied `OracleString` to a `string`.

Declaration

```
//C#
public static explicit operator string (OracleString value1);
```

Parameters

- *value1*

  The supplied `OracleString`.

Return Value

`string`

Exceptions

`OracleNullValueException` - The `OracleString` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

implicit operator OracleString

This type conversion operator converts the supplied `string` to an `OracleString`.

Declaration

```
// C#
public static implicit operator OracleString (string value1);
```

Parameters

- *value1*

  The supplied string.

Return Value

An `OracleString`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString Properties

The `OracleString` properties are listed in Table 5–85.

Table 5–85 OracleString Properties

Properties	Description
IsCaseIgnored	Indicates whether case should be ignored when performing string comparison
IsNull	Indicates whether the current instance has a null value
Item	Obtains the particular character in an `OracleString` using an index.
Length	Returns the length of the `OracleString`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

IsCaseIgnored

This property indicates whether case should be ignored when performing string comparison.

Declaration

```
//C#
public bool IsCaseIgnored {get;set;}
```

Property Value

Returns `true` if string comparison must ignore case; otherwise `false`.

Remarks

Default value is `true`.

Example

```
// C#
OracleString str1 = new OracleString("aAaAa");
OracleString str2 = new OracleString("AaAaA");

str1.IsCaseIgnored = true;
str2.IsCaseIgnored = true;

Console.WriteLine(str1.CompareTo(str2)); // Prints 0

// Note that IsCaseIgnored must be set to false for both OracleStrings
// otherwise an exception is thrown

str1.IsCaseIgnored = false;
str2.IsCaseIgnored = false;

Console.WriteLine(str1.CompareTo(str2)); // Prints non zero value
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

IsNull

This property indicates whether the current instance contains a null value.

Declaration

```
// C#
public bool IsNull {get;}
```

Property Value

Returns `true` if the current instance contains has a null value; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

Item

This property obtains the particular character in an `OracleString` using an index.

Declaration

```
// C#
public char Item {get;}
```

Property Value

A `char` value.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

Length

This property returns the length of the `OracleString`.

Declaration

```
// C#
public int Length {get;}
```

Property Value

A `int` value.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleString Methods

The `OracleString` methods are listed in Table 5–86.

Table 5–86 OracleString Methods

Methods	Description
Clone	Returns a copy of the current `OracleString` instance
CompareTo	Compares the current `OracleString` instance to the supplied object, and returns an integer that represents their relative values
Equals	Determines whether an object has the same string value as the current `OracleString` structure (Overloaded)
GetHashCode	Returns a hash code for the `OracleString` instance
GetNonUnicodeBytes	Returns an array of bytes, containing the contents of the `OracleString`, in the client character set format
GetType	Inherited from `Object`
GetUnicodeBytes	Returns an array of bytes, containing the contents of the `OracleString`, in Unicode format
ToString	Converts the current `OracleString` instance to a `string`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

Clone

This method creates a copy of an `OracleString` instance.

Declaration

```
// C#
public OracleString Clone();
```

Return Value

An `OracleString` structure.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
OracleString str_cloned = str.Clone();
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

CompareTo

This method compares the current `OracleString` instance to the supplied object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  The object being compared to the current instance.

Return Value

The method returns a number that is:

- Less than zero: if the current `OracleString` value is less than *obj*.
- Zero: if the current `OracleString` value is equal to *obj*.

- Greater than zero: if the current `OracleString` value is greater than *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* parameter is not of type `OracleString`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleString`s. For example, comparing an `OracleString` instance with an `OracleBinary` instance is not allowed. When an `OracleString` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

Equals

This method determines whether supplied object is an instance of `OracleString` and has the same values as the current `OracleString` instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *`obj`*

  An object being compared.

Return Value

Returns `true` if the supplied object is an instance of `OracleString` and has the same values as the current `OracleString` instance; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleString` that has a value is greater than an `OracleString` that has a null value.
- Two `OracleString`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleString` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

GetNonUnicodeBytes

This method returns an array of bytes, containing the contents of the `OracleString`, in the client character set format.

Declaration

```
// C#
public byte[] GetNonUnicodeBytes();
```

Return Value

A byte array that contains the contents of the `OracleString` in the client character set format.

Remarks

If the current instance has a null value, an `OracleNullValueException` is thrown.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

GetUnicodeBytes

This method returns an array of bytes, containing the contents of the `OracleString` in Unicode format.

Declaration

```
// C#
public byte[] GetUnicodeBytes();
```

Return Value

A byte array that contains the contents of the `OracleString` in Unicode format.

Remarks

If the current instance has a null value, an `OracleNullValueException` is thrown.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

ToString

Overrides `Object`

This method converts the current `OracleString` instance to a `string`.

Declaration

```
// C#
public override string ToString();
```

Return Value

A `string`.

Remarks

If the current `OracleString` instance has a null value, the string contains "null".

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleString Structure
- OracleString Members

OracleTimeStamp Structure

The `OracleTimeStamp` structure represents the Oracle `TIMESTAMP` datatype to be stored in or retrieved from a database. Each `OracleTimeStamp` stores the following information: year, month, day, hour, minute, second, and nanosecond.

Class Inheritance

```
Object
  ValueType
    OracleTimeStamp
```

Declaration

```
// C#
public struct OracleTimeStamp : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
// Illustrates usage of OracleTimeStamp

OracleTimeStamp  tsCurrent1 = OracleTimeStamp.GetSysDate();
OracleTimeStamp  tsCurrent2 = DateTime.Now;

// Calculate the difference between tsCurrent1 and tsCurrent2
OracleIntervalDS idsDiff = tsCurrent2.GetDaysBetween(tsCurrent1);

// Calculate the difference using AddNanoseconds()
int nanoDiff = 0;
while (tsCurrent2 > tsCurrent1)
{
  nanoDiff += 10;
  tsCurrent1 = tsCurrent1.AddNanoseconds(10);
}
Console.WriteLine("idsDiff.Nanoseconds = " + idsDiff.Nanoseconds);
      Console.WriteLine("nanoDiff = " + nanoDiff);
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Members
- OracleTimeStamp Constructors
- OracleTimeStamp Static Fields
- OracleTimeStamp Static Methods
- OracleTimeStamp Static Operators
- OracleTimeStamp Static Type Conversions
- OracleTimeStamp Properties
- OracleTimeStamp Methods

OracleTimeStamp Members

`OracleTimeStamp` members are listed in the following tables:

OracleTimeStamp Constructors

`OracleTimeStamp` constructors are listed in Table 5–87

Table 5–87 OracleTimeStamp Constructors

Constructor	Description
OracleTimeStamp Constructors	Instantiates a new instance of `OracleTimeStamp` structure (Overloaded)

OracleTimeStamp Static Fields

The `OracleTimeStamp` static fields are listed in Table 5–88.

Table 5–88 OracleTimeStamp Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStamp` structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStamp` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStamp` structure

OracleTimeStamp Static Methods

The `OracleTimeStamp` static methods are listed in Table 5–89.

Table 5–89 OracleTimeStamp Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStamp` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleTimeStamp` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStamp` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStamp` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStamp` values is less than or equal to the second
NotEquals	Determines if two `OracleTimeStamp` values are not equal
GetSysDate	Gets an `OracleTimeStamp` structure that represents the current date and time
Parse	Gets an `OracleTimeStamp` structure and sets its value using the supplied string

Table 5–89 OracleTimeStamp Static Methods (Cont.)

Methods	Description
SetPrecision	Returns a new instance of an `OracleTimeStamp` with the specified fractional second precision

OracleTimeStamp Static Operators

The `OracleTimeStamp` static operators are listed in Table 5–90.

Table 5–90 OracleTimeStamp Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure (Overloaded)
operator ==	Determines if two `OracleTimeStamp` values are equal
operator >	Determines if the first of two `OracleTimeStamp` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStamp` values is greater than or equal to the second
operator !=	Determines if the two `OracleTimeStamp` values are not equal
operator <	Determines if the first of two `OracleTimeStamp` values is less than the second
operator <=	Determines if the first of two `OracleTimeStamp` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure (Overloaded)

OracleTimeStamp Static Type Conversions

The `OracleTimeStamp` static type conversions are listed in Table 5–91.

Table 5–91 OracleTimeStamp Static Type Conversions

Operator	Description
explicit operator OracleTimeStamp	Converts an instance value to an `OracleTimeStamp` structure (Overloaded)
implicit operator OracleTimeStamp	Converts an instance value to an `OracleTimeStamp` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStamp` value to a `DateTime` structure

OracleTimeStamp Properties

The `OracleTimeStamp` properties are listed in Table 5–92.

Table 5–92 OracleTimeStamp Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle `TIMESTAMP` in Oracle internal format
Day	Specifies the day component of an `OracleTimeStamp`
IsNull	Indicates whether the `OracleTimeStamp` instance has a null value
Hour	Specifies the hour component of an `OracleTimeStamp`
Millisecond	Specifies the millisecond component of an `OracleTimeStamp`
Minute	Specifies the minute component of an `OracleTimeStamp`
Month	Specifies the month component of an `OracleTimeStamp`
Nanosecond	Specifies the nanosecond component of an `OracleTimeStamp`
Second	Specifies the second component of an `OracleTimeStamp`
Value	Specifies the date and time that is stored in the `OracleTimeStamp` structure
Year	Specifies the year component of an `OracleTimeStamp`

OracleTimeStamp Methods

The `OracleTimeStamp` methods are listed in Table 5–93.

Table 5–93 OracleTimeStamp Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStamp` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether an object has the same date and time as the current `OracleTimeStamp` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleTimeStamp` instance
GetDaysBetween	Subtracts an `OracleTimeStamp` value from the current instance and returns an `OracleIntervalDS` that represents the time difference between the supplied `OracleTimeStamp` and the current instance
GetYearsBetween	Subtracts `value1` from the current instance and returns an `OracleIntervalYM` that represents the difference between `value1` and the current instance using `OracleIntervalYM`

Table 5–93 OracleTimeStamp Methods (Cont.)

Methods	Description
GetType	Inherited from `Object`
ToOracleDate	Converts the current `OracleTimeStamp` structure to an `OracleDate` structure
ToOracleTimeStampLTZ	Converts the current `OracleTimeStamp` structure to an `OracleTimeStampLTZ` structure
ToOracleTimeStampTZ	Converts the current `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure
ToString	Converts the current `OracleTimeStamp` structure to a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure

OracleTimeStamp Constructors

The `OracleTimeStamp` constructors create new instances of the `OracleTimeStamp` structure.

Overload List:

- OracleTimeStamp(DateTime)

  This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using the supplied `DateTime` value.

- OracleTimeStamp(string)

  This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value using the supplied string.

- OracleTimeStamp(int, int, int)

  This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date using year, month, and day.

- OracleTimeStamp(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, and second.

- OracleTimeStamp(int, int, int, int, int, int, double)

  This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

- OracleTimeStamp(int, int, int, int, int, int, int)

  This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

- OracleTimeStamp(byte [ ])

  This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP` format.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(DateTime)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#
public OracleTimeStamp (DateTime dt);
```

Parameters

- *dt*

  The supplied `DateTime` value.

Exceptions

`ArgumentException` - The *`dt`* parameter cannot be used to construct a valid `OracleTimeStamp`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(string)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value using the supplied string.

Declaration

```
// C#
public OracleTimeStamp (string tsStr);
```

Parameters

- *`tsStr`*

  A string that represents an Oracle `TIMESTAMP`.

Exceptions

`ArgumentException` - The *`tsStr`* value is an invalid string representation of an Oracle `TIMESTAMP` or the supplied *`tsStr`* is not in the timestamp format specified by the `OracleGlobalization.TimeStampFormat` property of the thread, which represents Oracle's `NLS_TIMESTAMP_FORMAT` parameter.

`ArgumentNullException` - The *`tsStr`* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_format for the OracleTimeStamp(string) constructor
```

```
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStamp from a string using the format specified.

OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString() method
og.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(ts.ToString());
// Prints "1999-NOV-11 11:02:33.444000000 AM"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39
- *Oracle Database SQL Reference* for further information on date format elements

OracleTimeStamp(int, int, int)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date using year, month, and day.

Declaration

```
// C#
public OracleTimeStamp(int year, int month, int day);
```

Parameters

- *year*

  The year provided. Range of `year` is (-4712 to 9999).

- *month*

The month provided. Range of `month` is (1 to 12).

- *`day`*

  The day provided. Range of `day` is (1 to 31).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

```
// C#
public OracleTimeStamp (int year, int month, int day, int hour, int minute, int
second);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).

- *`month`*

  The month provided. Range of *`month`* is (1 to 12).

- *`day`*

  The day provided. Range of *`day`* is (1 to 31).

- *`hour`*

  The hour provided. Range of *`hour`* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(int, int, int, int, int, int, double)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

Declaration

```
// C#
public OracleTimeStamp(int year, int month, int day, int hour, int minute,
 int second, double millisecond);
```

Parameters

- *year*

  The year provided. Range of `year` is (-4712 to 9999).

- *month*

  The month provided. Range of `month` is (1 to 12).

- *day*

  The day provided. Range of `day` is (1 to 31).

- *hour*

The hour provided. Range of `hour` is (0 to 23).

- *`minute`*

  The minute provided. Range of `minute` is (0 to 59).

- *`second`*

  The second provided. Range of `second` is (0 to 59).

- *`milliSeconds`*

  The milliseconds provided. Range of `millisecond` is (0 to 999.999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(int, int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

```
// C#
public OracleTimeStamp (int year, int month, int day, int hour, int minute, int
second, int nanosecond);
```

Parameters

- *`year`*

  The year provided. Range of *`year`* is (-4712 to 9999).

- *`month`*

  The month provided. Range of *`month`* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

- *nanosecond*

  The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStamp` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(byte [ ])

This constructor creates a new instance of the `OracleTimeStamp` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP` format.

Declaration

```
// C#
public OracleTimeStamp (byte[] bytes);
```

Parameters

- *bytes*

A byte array that represents an Oracle TIMESTAMP in Oracle internal format.

Exceptions

ArgumentException - *bytes* is not in an internal Oracle TIMESTAMP format or *bytes* is not a valid Oracle TIMESTAMP.

ArgumentNullException - *bytes* is null.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp Static Fields

The OracleTimeStamp static fields are listed in Table 5–94.

Table 5–94 OracleTimeStamp Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an OracleTimeStamp structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an OracleTimeStamp structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the OracleTimeStamp structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

MaxValue

This static field represents the maximum valid date and time for an OracleTimeStamp structure, which is December 31, 9999 23:59:59.999999999.

Declaration

```
// C#
```

```
public static readonly OraTimestamp MaxValue;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

MinValue

This static field represents the minimum valid date and time for an `OracleTimeStamp` structure, which is January 1, -4712 0:0:0.

Declaration

```
// C#
public static readonly OracleTimeStamp MinValue;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Null

This static field represents a null value that can be assigned to an instance of the `OracleTimeStamp` structure.

Declaration

```
// C#
public static readonly OracleTimeStamp Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp Static Methods

The `OracleTimeStamp` static methods are listed in Table 5–95.

Table 5–95 OracleTimeStamp Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStamp` values are equal (Overloaded)
GreaterThan	Determines if the first of two `OracleTimeStamp` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStamp` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStamp` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStamp` values is less than or equal to the second
NotEquals	Determines if two `OracleTimeStamp` values are not equal
GetSysDate	Gets an `OracleTimeStamp` structure that represents the current date and time
Parse	Gets an `OracleTimeStamp` structure and sets its value using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStamp` with the specified fractional second precision

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Equals

This static method determines if two `OracleTimeStamp` values are equal.

Declaration

```
// C#
public static bool Equals(OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second OracleTimeStamp.

Return Value

Returns true if two OracleTimeStamp values are equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimeStamps that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

GreaterThan

This static method determines if the first of two OracleTimeStamp values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First OracleTimeStamp.

- *value2*

  Second OracleTimeStamp.

Return Value

Returns true if the first of two OracleTimeStamp values is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

GreaterThanOrEqual

This static method determines if the first of two `OracleTimeStamp` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleTimeStamp value1, OracleTimeStamp
value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if the first of two `OracleTimeStamp` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

LessThan

This static method determines if the first of two `OracleTimeStamp` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if the first of two `OracleTimeStamp` values is less than the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

LessThanOrEqual

This static method determines if the first of two `OracleTimeStamp` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleTimeStamp value1, OracleTimeStamp
value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if the first of two `OracleTimeStamp` values is less than or equal to the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

NotEquals

This static method determines if two `OracleTimeStamp` values are not equal.

Declaration

```
// C#
```

```
public static bool NotEquals(OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First OracleTimeStamp.

- *value2*

  Second OracleTimeStamp.

Return Value

Returns true if two OracleTimeStamp values are not equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.
- Two OracleTimeStamps that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

GetSysDate

This static method gets an OracleTimeStamp structure that represents the current date and time.

Declaration

```
// C#
public static OracleTimeStamp GetSysDate();
```

Return Value

An OracleTimeStamp structure that represents the current date and time.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Parse

This static method gets an `OracleTimeStamp` structure and sets its value using the supplied string.

Declaration

```
// C#
public static OracleTimeStamp Parse(string datetime);
```

Parameters

- *`datetime`*

  A string that represents an Oracle `TIMESTAMP`.

Return Value

An `OracleTimeStamp` structure.

Exceptions

`ArgumentException` - The *`tsStr`* is an invalid string representation of an Oracle `TIMESTAMP` or the supplied *`tsStr`* is not in the timestamp format specified by the `OracleGlobalization.TimeStampFormat` property of the thread, which represents Oracle's `NLS_TIMESTAMP_FORMAT` parameter.

`ArgumentNullException` - The *`tsStr`* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_format for the Parse() method
```

```
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStamp from a string using the format specified.

OracleTimeStamp ts = OracleTimeStamp.Parse("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString() method
og.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(ts.ToString());
// Prints "1999-NOV-11 11:02:33.444000000 AM"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

SetPrecision

This static method returns a new instance of an `OracleTimeStamp` with the specified fractional second precision.

Declaration

```
// C#
public static OracleTimeStamp SetPrecision(OracleTimeStamp value1, int
fracSecPrecision);
```

Parameters

- *value1*

  The provided `OracleTimeStamp` object.

- *fracSecPrecision*

  The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleTimeStamp` structure with the specified fractional second precision.

Exceptions

`ArgumentOutOfRangeException` - *`fracSecPrecision`* is out of the specified range.

Remarks

The value specified in the supplied *`fracSecPrecision`* is used to perform a rounding off operation on the supplied `OracleTimeStamp` value. Depending on this value, `0` or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleTimeStamp` with a value of "`December 31, 9999 23:59:59.99`" results in the string "`December 31, 9999 23:59:59.99000`" when `SetPrecision()` is called with the fractional second precision set to `5`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp Static Operators

The `OracleTimeStamp` static operators are listed in Table 5–96.

Table 5–96 OracleTimeStamp Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure (Overloaded)
operator ==	Determines if two `OracleTimeStamp` values are equal
operator >	Determines if the first of two `OracleTimeStamp` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStamp` values is greater than or equal to the second

Table 5–96 OracleTimeStamp Static Operators (Cont.)

Operator	Description
operator !=	Determines if the two `OracleTimeStamp` values are not equal
operator <	Determines if the first of two `OracleTimeStamp` values is less than the second
operator <=	Determines if the first of two `OracleTimeStamp` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator +

`operator+` adds the supplied object to the `OracleTimeStamp` and returns a new `OracleTimeStamp` structure.

Overload List:

- operator + (OracleTimeStamp, OracleIntervalDS)

  This static operator adds the supplied `OracleIntervalDS` to the `OracleTimeStamp` and returns a new `OracleTimeStamp` structure.

- operator + (OracleTimeStamp, OracleIntervalYM)

  This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure.

- operator + (OracleTimeStamp, TimeSpan)

  This static operator adds the supplied `TimeSpan` to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator + (OracleTimeStamp, OracleIntervalDS)

This static operator adds the supplied `OracleIntervalDS` to the `OracleTimeStamp` and returns a new `OracleTimeStamp` structure.

Declaration

```
// C#
public static operator + (OracleTimeStamp value1, OracleIntervalDS value2);
```

Parameters

- *value1*

  An `OracleTimeStamp`.

- *value2*

  An `OracleIntervalDS`.

Return Value

An `OracleTimeStamp`.

Remarks

If either parameter has a null value, the returned `OracleTimeStamp` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator + (OracleTimeStamp, OracleIntervalYM)

This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure.

Declaration

```
// C#
public static operator + (OracleTimeStamp value1, OracleIntervalYM value2);
```

Parameters

- *value1*

  An `OracleTimeStamp`.

- *value2*

  An `OracleIntervalYM`.

Return Value

An `OracleTimeStamp`.

Remarks

If either parameter has a null value, the returned `OracleTimeStamp` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator + (OracleTimeStamp, TimeSpan)

This static operator adds the supplied `TimeSpan` to the supplied `OracleTimeStamp` and returns a new `OracleTimeStamp` structure.

Declaration

```
// C#
public static operator + (OracleTimeStamp value1, TimeSpan value2);
```

Parameters

- *value1*

  An `OracleTimeStamp`.

- *value2*

  A `TimeSpan`.

Return Value

An `OracleTimeStamp`.

Remarks

If the `OracleTimeStamp` instance has a null value, the returned `OracleTimeStamp` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator ==

This static operator determines if two `OracleTimeStamp` values are equal.

Declaration

```
// C#
public static bool operator == (OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if they are the same; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator >

This static operator determines if the first of two `OracleTimeStamp` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if the first `OracleTimeStamp` value is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator >=

This static operator determines if the first of two `OracleTimeStamp` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if the first `OracleTimeStamp` is greater than or equal to the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator !=

This static operator determines if two `OracleTimeStamp` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if two `OracleTimeStamp` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator <

This static operator determines if the first of two `OracleTimeStamp` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if the first `OracleTimeStamp` is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator <=

This static operator determines if the first of two `OracleTimeStamp` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- *value1*

  First `OracleTimeStamp`.

- *value2*

  Second `OracleTimeStamp`.

Return Value

Returns `true` if the first `OracleTimeStamp` is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator -

`operator-` subtracts the supplied value, from the supplied `OracleTimeStamp` value, and returns a new `OracleTimeStamp` structure.

Overload List:

- operator - (OracleTimeStamp, OracleIntervalDS)

  This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStamp` value, and return a new `OracleTimeStamp` structure.

- operator - (OracleTimeStamp, OracleIntervalYM)

  This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStamp` value, and returns a new `OracleTimeStamp` structure.

- operator - (OracleTimeStamp, TimeSpan)

  This static operator subtracts the supplied `TimeSpan` value, from the supplied `OracleTimeStamp` value, and returns a new `OracleTimeStamp` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator - (OracleTimeStamp, OracleIntervalDS)

This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStamp` value, and return a new `OracleTimeStamp` structure.

Declaration

```
// C#
public static operator - (OracleTimeStamp value1, OracleIntervalDS value2);
```

Parameters

- *value1*

  An `OracleTimeStamp`.

- *value2*

  An `OracleIntervalDS` instance.

Return Value

An `OracleTimeStamp` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStamp` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator - (OracleTimeStamp, OracleIntervalYM)

This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStamp` value, and returns a new `OracleTimeStamp` structure.

Declaration

```
// C#
public static operator - (OracleTimeStamp value1, OracleIntervalYM value2);
```

Parameters

- *value1*

  An OracleTimeStamp.

- *value2*

  An OracleIntervalYM instance.

Return Value

An OracleTimeStamp structure.

Remarks

If either parameter has a null value, the returned OracleTimeStamp has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator - (OracleTimeStamp, TimeSpan)

This static operator subtracts the supplied TimeSpan value, from the supplied OracleTimeStamp value, and returns a new OracleTimeStamp structure.

Declaration

```
// C#
public static operator - (OracleTimeStamp value1, TimeSpan value2);
```

Parameters

- *value1*

  An OracleTimeStamp.

- *value2*

  A TimeSpan instance.

Return Value

An OracleTimeStamp structure.

Remarks

If the `OracleTimeStamp` instance has a null value, the returned `OracleTimeStamp` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp Static Type Conversions

The `OracleTimeStamp` static type conversions are listed in Table 5–97.

Table 5–97 OracleTimeStamp Static Type Conversions

Operator	Description
explicit operator OracleTimeStamp	Converts an instance value to an `OracleTimeStamp` structure (Overloaded)
implicit operator OracleTimeStamp	Converts an instance value to an `OracleTimeStamp` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStamp` value to a `DateTime` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

explicit operator OracleTimeStamp

`explicit operator OracleTimeStamp` converts the supplied value to an `OracleTimeStamp` structure

Overload List:

- explicit operator OracleTimeStamp(OracleTimeStampLTZ)

  This static type conversion operator converts an `OracleTimeStampLTZ` value to an `OracleTimeStamp` structure.

- explicit operator OracleTimeStamp(OracleTimeStampTZ)

  This static type conversion operator converts an `OracleTimeStampTZ` value to an `OracleTimeStamp` structure.

- explicit operator OracleTimeStamp(string)

  This static type conversion operator converts the supplied string to an `OracleTimeStamp` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

explicit operator OracleTimeStamp(OracleTimeStampLTZ)

This static type conversion operator converts an `OracleTimeStampLTZ` value to an `OracleTimeStamp` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStamp(OracleTimeStampLTZ value1);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ` instance.

Return Value

The returned `OracleTimeStamp` contains the date and time of the `OracleTimeStampLTZ` structure.

Remarks

If the `OracleTimeStampLTZ` structure has a null value, the returned `OracleTimeStamp` structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

explicit operator OracleTimeStamp(OracleTimeStampTZ)

This static type conversion operator converts an `OracleTimeStampTZ` value to an `OracleTimeStamp` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStamp(OracleTimeStampTZ value1);
```

Parameters

- *value1*

  An `OracleTimeStampTZ` instance.

Return Value

The returned `OracleTimeStamp` contains the date and time information from *value1*, but the time zone information from *value1* is truncated.

Remarks

If the `OracleTimeStampTZ` structure has a null value, the returned `OracleTimeStamp` structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

explicit operator OracleTimeStamp(string)

This static type conversion operator converts the supplied string to an `OracleTimeStamp` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStamp(string tsStr);
```

Parameters

- *tsStr*

  A string representation of an Oracle TIMESTAMP.

Return Value

A OracleTimeStamp.

Exceptions

ArgumentException - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP or the *tsStr* is not in the timestamp format specified by the thread's OracleGlobalization.TimeStampFormat property, which represents Oracle's NLS_TIMESTAMP_FORMAT parameter.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_format for the explicit operator //
OracleTimeStamp(string)
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStamp from a string using the format specified.

OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString method
og.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(ts.ToString());
```

```
// Prints "1999-NOV-11 11:02:33.444000000 AM"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39
- *Oracle Database SQL Reference* for further information on datetime format elements

implicit operator OracleTimeStamp

This static type conversion operator converts a value to an `OracleTimeStamp` structure.

Overload List:

- implicit operator OracleTimeStamp(OracleDate)

  This static type conversion operator converts an `OracleDate` value to an `OracleTimeStamp` structure.

- implicit operator OracleTimeStamp(DateTime)

  This static type conversion operator converts a `DateTime` value to an `OracleTimeStamp` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

implicit operator OracleTimeStamp(OracleDate)

This static type conversion operator converts an `OracleDate` value to an `OracleTimeStamp` structure.

Declaration

```
// C#
public static implicit operator OracleTimeStamp (OracleDate value1);
```

Parameters

- *value1*

  An OracleDate instance.

Return Value

An OracleTimeStamp structure that contains the date and time of the OracleDate structure, *value1*.

Remarks

If the OracleDate structure has a null value, the returned OracleTimeStamp structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

implicit operator OracleTimeStamp(DateTime)

This static type conversion operator converts a DateTime value to an OracleTimeStamp structure.

Declaration

```
// C#
public static implicit operator OracleTimeStamp(DateTime value);
```

Parameters

- *value*

  A DateTime instance.

Return Value

An OracleTimeStamp structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

explicit operator DateTime

This static type conversion operator converts an `OracleTimeStamp` value to a `DateTime` structure.

Declaration

```
// C#
public static explicit operator DateTime(OracleTimeStamp value1);
```

Parameters

- *value1*

  An `OracleTimeStamp` instance.

Return Value

A `DateTime` containing the date and time in the current instance.

Exceptions

`OracleNullValueException` - The `OracleTimeStamp` structure has a null value.

Remarks

The precision of the `OracleTimeStamp` can be lost during the conversion.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp Properties

The `OracleTimeStamp` properties are listed in Table 5–98.

Table 5–98 OracleTimeStamp Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle `TIMESTAMP` in Oracle internal format
Day	Specifies the day component of an `OracleTimeStamp`
IsNull	Indicates whether the `OracleTimeStamp` instance has a null value
Hour	Specifies the hour component of an `OracleTimeStamp`
Millisecond	Specifies the millisecond component of an `OracleTimeStamp`
Minute	Specifies the minute component of an `OracleTimeStamp`
Month	Specifies the month component of an `OracleTimeStamp`
Nanosecond	Specifies the nanosecond component of an `OracleTimeStamp`
Second	Specifies the second component of an `OracleTimeStamp`
Value	Specifies the date and time that is stored in the `OracleTimeStamp` structure
Year	Specifies the year component of an `OracleTimeStamp`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

BinData

This property returns an array of bytes that represents an Oracle `TIMESTAMP` in Oracle internal format.

Declaration

```
// C#
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle `TIMESTAMP` in an internal format.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Day

This property specifies the day component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Day{get;}
```

Property Value

A number that represents the day. Range of `Day` is (1 to 31).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

IsNull

This property indicates whether the current instance has a null value.

Declaration

```
// C#
public bool IsNull{get;}
```

Property Value

Returns `true` if the current instance has a null value; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Hour

This property specifies the hour component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of `hour` is (0 to 23).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Millisecond

This property gets the millisecond component of an `OracleTimeStamp`.

Declaration

```
// C#
public double Millisecond{get;}
```

Property Value

A number that represents a millisecond. Range of `Millisecond` is (0 to 999.999999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Minute

This property gets the minute component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Minute{get;}
```

Property Value

A number that represent a minute. Range of `Minute` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Month

This property gets the month component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Month{get;}
```

Property Value

A number that represents a month. Range of `Month` is (1 to 12).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Nanosecond

This property gets the nanosecond component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Nanosecond{get;}
```

Property Value

A number that represents a nanosecond. Range of `Nanosecond` is (0 to 999999999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Second

This property gets the second component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Second{get;}
```

Property Value

A number that represents a second. Range of `Second` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Value

This property specifies the date and time that is stored in the `OracleTimeStamp` structure.

Declaration

```
// C#
public DateTime Value{get;}
```

Property Value

A `DateTime`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Year

This property gets the year component of an `OracleTimeStamp`.

Declaration

```
// C#
public int Year{get;}
```

Property Value

A number that represents a year. The range of `Year` is (-4712 to 9999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp Methods

The `OracleTimeStamp` methods are listed in Table 5–99.

Table 5–99 OracleTimeStamp Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStamp` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether an object has the same date and time as the current `OracleTimeStamp` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleTimeStamp` instance
GetDaysBetween	Subtracts an `OracleTimeStamp` value from the current instance and returns an `OracleIntervalDS` that represents the time difference between the supplied `OracleTimeStamp` and the current instance

Table 5–99 OracleTimeStamp Methods (Cont.)

Methods	Description
GetYearsBetween	Subtracts `value1` from the current instance and returns an `OracleIntervalYM` that represents the difference between `value1` and the current instance using `OracleIntervalYM`
GetType	Inherited from `Object`
ToOracleDate	Converts the current `OracleTimeStamp` structure to an `OracleDate` structure
ToOracleTimeStampLTZ	Converts the current `OracleTimeStamp` structure to an `OracleTimeStampLTZ` structure
ToOracleTimeStampTZ	Converts the current `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure
ToString	Converts the current `OracleTimeStamp` structure to a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddDays

This method adds the supplied number of days to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddDays(double days);
```

Parameters

- *days*

  The supplied number of days. Range is (-1,000,000,000 < *days* < 1,000,000,000)

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddHours(double hours);
```

Parameters

- *hours*

  The supplied number of hours. Range is (-24,000,000,000 < *hours* < 24,000,000,000).

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddMilliseconds(double milliseconds);
```

Parameters

- *milliseconds*

  The supplied number of milliseconds. Range is (-8.64 * 1016< `milliseconds` < 8.64 * 1016).

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddMinutes(double minutes);
```

Parameters

- *minutes*

  The supplied number of minutes. Range is (-1,440,000,000,000 < *minutes* < 1,440,000,000,000).

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddMonths

This method adds the supplied number of months to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddMonths(long months);
```

Parameters

- *months*

  The supplied number of months. Range is (-12,000,000,000 < *months* < 12,000,000,000).

Return Value

An `OracleTimeStamp`.

Exceptions

`ArgumentOutofRangeException` - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddNanoseconds(long nanoseconds);
```

Parameters

- *nanoseconds*

  The supplied number of nanoseconds.

Return Value

An `OracleTimeStamp`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddSeconds

This method adds the supplied number of seconds to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddSeconds(double seconds);
```

Parameters

- *seconds*

  The supplied number of seconds. Range is (-8.64 * 1013< seconds < 8.64 * 1013).

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutofRangeException - The argument value is out of the specified range.

OracleNullValueException - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddYears

This method adds the supplied number of years to the current instance.

Declaration

```
// C#
public OracleTimeStamp AddYears(int years);
```

Parameters

- *years*

  The supplied number of years. Range is (-999,999,999 <= *years* < = 999,999,999)

Return Value

An OracleTimeStamp.

Exceptions

ArgumentOutofRangeException - The argument value is out of the specified range.

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

CompareTo

This method compares the current `OracleTimeStamp` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  The object being compared to the current `OracleTimeStamp` instance.

Return Value

The method returns a number that is:

Less than zero: if the current `OracleTimeStamp` instance value is less than that of *obj*.

Zero: if the current `OracleTimeStamp` instance and *obj* values are equal.

Greater than zero: if the current `OracleTimeStamp` instance value is greater than that of *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *obj* parameter is not of type `OracleTimeStamp`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleTimeStamps`. For example, comparing an `OracleTimeStamp` instance with an `OracleBinary` instance is not allowed. When an `OracleTimeStamp` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

Equals

Overrides `Object`

This method determines whether an object has the same date and time as the current `OracleTimeStamp` instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *obj*

  The object being compared to the current `OracleTimeStamp` instance.

Return Value

Returns `true` if the *obj* is of type `OracleTimeStamp` and represents the same date and time; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamps` that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleTimeStamp` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

GetDaysBetween

This method subtracts an `OracleTimeStamp` value from the current instance and returns an `OracleIntervalDS` that represents the time difference between the supplied `OracleTimeStamp` structure and the current instance.

Declaration

```
// C#
public OracleIntervalDS GetDaysBetween(OracleTimeStamp value1);
```

Parameters

- *`value1`*

  The `OracleTimeStamp` value being subtracted.

Return Value

An `OracleIntervalDS` that represents the interval between two `OracleTimeStamp` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalDS` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

GetYearsBetween

This method subtracts an `OracleTimeStamp` value from the current instance and returns an `OracleIntervalYM` that represents the time difference between the `OracleTimeStamp` value and the current instance.

Declaration

```
// C#
public OracleIntervalYM GetYearsBetween(OracleTimeStamp value1);
```

Parameters

- *value1*

  The `OracleTimeStamp` value being subtracted.

Return Value

An `OracleIntervalYM` that represents the interval between two `OracleTimeStamp` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalYM` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

ToOracleDate

This method converts the current `OracleTimeStamp` structure to an `OracleDate` structure.

Declaration

```
// C#
public OracleDate ToOracleDate();
```

Return Value

The returned `OracleDate` contains the date and time in the current instance.

Remarks

The precision of the `OracleTimeStamp` value can be lost during the conversion.

If the value of the `OracleTimeStamp` has a null value, the value of the returned `OracleDate` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

ToOracleTimeStampLTZ

This method converts the current `OracleTimeStamp` structure to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public OracleTimeStampLTZ ToOracleTimeStampLTZ();
```

Return Value

The returned `OracleTimeStampLTZ` contains date and time in the current instance.

Remarks

If the value of the current instance has a null value, the value of the returned `OracleTimeStampLTZ` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members

ToOracleTimeStampTZ

This method converts the current `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public OracleTimeStampTZ ToOracleTimeStampTZ();
```

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleTimeStamp` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

If the value of the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

ToString

Overrides `Object`

This method converts the current `OracleTimeStamp` structure to a string.

Declaration

```
// C#
public override string ToString();
```

Return Value

A `string` that represents the same date and time as the current `OracleTimeStamp` structure.

Remarks

The returned value is a string representation of an `OracleTimeStamp` in the format specified by the `OracleGlobalization.TimeStampFormat` property of the thread.

The names and abbreviations used for months and days are in the language specified by the `OracleGlobalization`'s `DateLanguage` and `Calendar` properties of the thread. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_format for the OracleTimeStamp(string)constructor
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStamp from a string using the format specified.

OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString() method
og.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(ts.ToString());
// Prints "1999-NOV-11 11:02:33.444000000 AM"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

OracleTimeStampLTZ Structure

The `OracleTimeStampLTZ` structure represents the Oracle `TIMESTAMP WITH LOCAL TIME ZONE` data type to be stored in or retrieved from a database. Each `OracleTimeStampLTZ` stores the following information: year, month, day, hour, minute, second, and nanosecond.

Class Inheritance

```
Object

  ValueType

    OracleTimeStampLTZ
```

Declaration

```
// C#
public struct OracleTimeStampLTZ : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
// Illustrates usage of OracleTimeStampLTZ
// Display Local Time Zone Name
Console.WriteLine("Local Time Zone Name = " +
     OracleTimeStampLTZ.GetLocalTimeZoneName());

OracleTimeStampLTZ  tsLocal1 = OracleTimeStampLTZ.GetSysDate();
OracleTimeStampLTZ  tsLocal2 = DateTime.Now;

// Calculate the difference between tsLocal1 and tsLocal2
OracleIntervalDS idsDiff = tsLocal2.GetDaysBetween(tsLocal1);

// Calculate the difference using AddNanoseconds()
int nanoDiff = 0;
while (tsLocal2 > tsLocal1)
{
   nanoDiff += 10;
   tsLocal1 = tsLocal1.AddNanoseconds(10);
}
```

```
Console.WriteLine("idsDiff.Nanoseconds = " + idsDiff.Nanoseconds);
Console.WriteLine("nanoDiff = " + nanoDiff);
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Members
- OracleTimeStampLTZ Constructors
- OracleTimeStampLTZ Static Fields
- OracleTimeStampLTZ Static Methods
- OracleTimeStampLTZ Static Type Operators
- OracleTimeStampLTZ Static Type Conversions
- OracleTimeStampLTZ Properties
- OracleTimeStampLTZ Methods

OracleTimeStampLTZ Members

`OracleTimeStampLTZ` members are listed in the following tables:

OracleTimeStampLTZ Constructors

`OracleTimeStampLTZ` constructors are listed in Table 5–100

Table 5–100 OracleTimeStampLTZConstructors

Constructor	Description
OracleTimeStampLTZ Constructors	Instantiates a new instance of `OracleTimeStampLTZ` structure (Overloaded)

OracleTimeStampLTZ Static Fields

The `OracleTimeStampLTZ` static fields are listed in Table 5–101.

Table 5–101 OracleTimeStampLTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStampLTZ` structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStampLTZ` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStampLTZ` structure

OracleTimeStampLTZ Static Methods

The `OracleTimeStampLTZ` static methods are listed in Table 5–102.

Table 5–102 OracleTimeStampLTZ Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStampLTZ` values are equal (Overloaded)
GetLocalTimeZoneName	Gets the client's local time zone name
GetLocalTimeZoneOffset	Gets the client's local time zone offset relative to UTC
GetSysDate	Gets an `OracleTimeStampLTZ` structure that represents the current date and time
GreaterThan	Determines if the first of two `OracleTimeStampLTZ` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStampLTZ` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second

Table 5–102 OracleTimeStampLTZ Static Methods (Cont.)

Methods	Description
NotEquals	Determines if two `OracleTimeStampLTZ` values are not equal
Parse	Gets an `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStampLTZ` with the specified fractional second precision

OracleTimeStampLTZ Static Type Operators

The `OracleTimeStampLTZ` static type operators are listed in Table 5–103.

Table 5–103 OracleTimeStampLTZ Static Operators

Operator	Description
operator+	Adds the supplied instance value to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure (Overloaded)
operator ==	Determines if two `OracleTimeStampLTZ` values are equal
operator >	Determines if the first of two `OracleTimeStampLTZ` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second
operator !=	Determines if two `OracleTimeStampLTZ` values are not equal
operator <	Determines if the first of two `OracleTimeStampLTZ` values is less than the second
operator <=	Determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second

Table 5–103 OracleTimeStampLTZ Static Operators (Cont.)

Operator	Description
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure (Overloaded)

OracleTimeStampLTZ Static Type Conversions

The `OracleTimeStampLTZ` static type conversions are listed in Table 5–104.

Table 5–104 OracleTimeStampLTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampLTZ	Converts an instance value to an `OracleTimeStampLTZ` structure (Overloaded)
implicit operator OracleTimeStampLTZ	Converts an instance value to an `OracleTimeStampLTZ` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStampLTZ` value to a `DateTime` structure

OracleTimeStampLTZ Properties

The `OracleTimeStampLTZ` properties are listed in Table 5–105.

Table 5–105 OracleTimeStampLTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an `Oracle TIMESTAMP WITH LOCAL TIME ZONE` in Oracle internal format
Day	Specifies the `day` component of an `OracleTimeStampLTZ`
IsNull	Indicates whether the `OracleTimeStampLTZ` instance has a null value
Hour	Specifies the hour component of an `OracleTimeStampLTZ`
Millisecond	Specifies the millisecond component of an `OracleTimeStampLTZ`

Table 5–105 OracleTimeStampLTZ Properties (Cont.)

Properties	Description
Minute	Specifies the minute component of an `OracleTimeStampLTZ`
Month	Specifies the month component of an `OracleTimeStampLTZ`
Nanosecond	Specifies the nanosecond component of an `OracleTimeStampLTZ`
Second	Specifies the second component of an `OracleTimeStampLTZ`
Value	Specifies the date and time that is stored in the `OracleTimeStampLTZ` structure
Year	Specifies the year component of an `OracleTimeStampLTZ`

OracleTimeStampLTZ Methods

The `OracleTimeStampLTZ` methods are listed in Table 5–106.

Table 5–106 OracleTimeStampLTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance

Table 5–106 OracleTimeStampLTZ Methods (Cont.)

Methods	Description
CompareTo	Compares the current `OracleTimeStampLTZ` instance to an object and returns an integer that represents their relative values
Equals	Determines whether an object has the same date and time as the current `OracleTimeStampLTZ` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleTimeStampLTZ` instance
GetDaysBetween	Subtracts an `OracleTimeStampLTZ` from the current instance and returns an `OracleIntervalDS` that represents the difference
GetYearsBetween	Subtracts an `OracleTimeStampLTZ` from the current instance and returns an `OracleIntervalYM` that represents the difference
GetType	Inherited from `Object`
ToOracleDate	Converts the current `OracleTimeStampLTZ` structure to an `OracleDate` structure
ToOracleTimeStamp	Converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStamp` structure
ToOracleTimeStampTZ	Converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStampTZ` structure
ToString	Converts the current `OracleTimeStampLTZ` structure to a string
ToUniversalTime	Converts the current local time to Coordinated Universal Time (UTC)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure

OracleTimeStampLTZ Constructors

The `OracleTimeStampLTZ` constructors create new instances of the `OracleTimeStampLTZ` structure.

Overload List:

- OracleTimeStampLTZ(DateTime)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied `DateTime` value.

- OracleTimeStampLTZ(string)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string.

- OracleTimeStampLTZ(int, int, int)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date using year, month, and day.

- OracleTimeStampLTZ(int, int, int, int, int, int)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.

- OracleTimeStampLTZ(int, int, int, int, int, int, double)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

- OracleTimeStampLTZ(int, int, int, int, int, int, int)

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

- OracleTimeStampLTZ(byte [ ])

  This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP WITH LOCAL TIME ZONE` format.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ(DateTime)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#
public OracleTimeStampLTZ (DateTime dt);
```

Parameters

- *dt*

  The supplied `DateTime` value.

Exceptions

`ArgumentException` - The *dt* parameter cannot be used to construct a valid `OracleTimeStampLTZ`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ(string)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#
public OracleTimeStampLTZ(string tsStr);
```

Parameters

- *tsStr*

A string that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE.

Exceptions

ArgumentException - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP WITH LOCAL TIME ZONE or the supplied *tsStr* is not in the timestamp format specified by the OracleGlobalization.TimeStampFormat property of the thread, which represents Oracle's NLS_TIMESTAMP_FORMAT parameter.

ArgumentNullException - The *tsStr* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_format for the OracleTimeStampLTZ(string) constructor
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStampLTZ from a string using the format specified.

OracleTimeStampLTZ ts=new OracleTimeStampLTZ("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString() method
og.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(ts.ToString()); // Prints "1999-NOV-11 11:02:33.444000000 AM"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39
- *Oracle Database SQL Reference* for further information on date format elements

OracleTimeStampLTZ(int, int, int)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date using year, month, and day.

Declaration

```
// C#
public OracleTimeStampLTZ(int year, int month, int day);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

```
// C#
public OracleTimeStampLTZ (int year, int month, int day, int hour, int minute,
int second);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ(int, int, int, int, int, int, double)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

Declaration

```
// C#
public OracleTimeStampLTZ(int year, int month, int day, int hour, int minute,
 int second, double millisecond);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

- *milliSeconds*

The milliseconds provided. Range of *millisecond* is (0 to 999.999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ(int, int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

```
// C#
public OracleTimeStampLTZ (int year, int month, int day, int hour, int minute,
int second, int nanosecond);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

- *nanosecond*

  The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ(byte [ ])

This constructor creates a new instance of the `OracleTimeStampLTZ` structure and sets its value to the provided byte array, which is in the internal Oracle `TIMESTAMP WITH LOCAL TIME ZONE` format.

Declaration

```
// C#
public OracleTimeStampLTZ (byte[] bytes);
```

Parameters

- *bytes*

  A byte array that represents an Oracle `TIMESTAMP WITH LOCAL TIME ZONE` in Oracle internal format.

Exceptions

`ArgumentException` - *`bytes`* is not in an internal Oracle `TIMESTAMP WITH LOCAL TIME ZONE` format or *`bytes`* is not a valid Oracle `TIMESTAMP WITH LOCAL TIME ZONE`.

`ArgumentNullException` - *`bytes`* is null.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ Static Fields

The `OracleTimeStampLTZ` static fields are listed in Table 5–107.

Table 5–107 OracleTimeStampLTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStampLTZ` structure, which is December 31, 9999 23:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStampLTZ` structure, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStampLTZ` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

MaxValue

This static field represents the maximum valid date for an `OracleTimeStampLTZ` structure, which is December 31, 9999 23:59:59.999999999.

Declaration

```
// C#
public static readonly OracleTimeStampLTZ MaxValue;
```

Remarks

This value is the maximum date and time in the client time zone.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

MinValue

This static field represents the minimum valid date for an `OracleTimeStampLTZ` structure, which is January 1, -4712 0:0:0.

Declaration

```
// C#
public static readonly OracleTimeStampLTZ MinValue;
```

Remarks

This value is the minimum date and time in the client time zone.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Null

This static field represents a null value that can be assigned to an instance of the `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static readonly OracleTimeStampLTZ Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ Static Methods

The `OracleTimeStampLTZ` static methods are listed in Table 5–108.

Table 5–108 OracleTimeStampLTZ Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStampLTZ` values are equal (Overloaded)
GetLocalTimeZoneName	Gets the client's local time zone name
GetLocalTimeZoneOffset	Gets the client's local time zone offset relative to UTC
GetSysDate	Gets an `OracleTimeStampLTZ` structure that represents the current date and time
GreaterThan	Determines if the first of two `OracleTimeStampLTZ` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStampLTZ` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second
NotEquals	Determines if two `OracleTimeStampLTZ` values are not equal
Parse	Gets an `OracleTimeStampLTZ` structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStampLTZ` with the specified fractional second precision

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Equals

This static method determines if two `OracleTimeStampLTZ` values are equal.

Declaration

```
// C#
public static bool Equals(OracleTimeStampLTZ value1, OracleTimeStampLTZ value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if two `OracleTimeStampLTZ` values are equal. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GetLocalTimeZoneName

This static method gets the client's local time zone name.

Declaration

```
// C#
public static string GetLocalTimeZoneName();
```

Return Value

A string containing the local time zone.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GetLocalTimeZoneOffset

This static method gets the client's local time zone offset relative to Coordinated Universal Time (UTC).

Declaration

```
// C#
public static TimeSpan OracleTimeStampLTZ GetLocalTimeZoneOffset( );
```

Return Value

A `TimeSpan` structure containing the local time zone hours and time zone minutes.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GetSysDate

This static method gets an `OracleTimeStampLTZ` structure that represents the current date and time.

Declaration

```
// C#
public static OracleTimeStampLTZ GetSysDate();
```

Return Value

An `OracleTimeStampLTZ` structure that represents the current date and time.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GreaterThan

This static method determines if the first of two `OracleTimeStampLTZ` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampLTZ` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.

- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GreaterThanOrEqual

This static method determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleTimeStampLTZ value1,
OracleTimeStampLTZ value2);
```

Parameters

- *`value1`*

  First `OracleTimeStampLTZ`.

- *`value2`*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

LessThan

This static method determines if the first of two `OracleTimeStampLTZ` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampLTZ` values is less than the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

LessThanOrEqual

This static method determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampLTZ` values is less than or equal to the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

NotEquals

This static method determines if two `OracleTimeStampLTZ` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleTimeStampLTZ value1, OracleTimeStampLTZ
```

```
value2);
```

Parameters

- *value1*

  First OracleTimeStampLTZ.

- *value2*

  Second OracleTimeStampLTZ.

Return Value

Returns true if two OracleTimeStampLTZ values are not equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZs that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Parse

This static method creates an OracleTimeStampLTZ structure and sets its value using the supplied string.

Declaration

```
// C#
public static OracleTimeStampLTZ Parse(string tsStr);
```

Parameters

- *tsStr*

  A string that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE.

Return Value

An `OracleTimeStampLTZ` structure.

Exceptions

`ArgumentException` - The *`tsStr`* parameter is an invalid string representation of an Oracle `TIMESTAMP WITH LOCAL TIME ZONE` or the *`tsStr`* is not in the timestamp format specified by the `OracleGlobalization.TimeStampFormat` property of the thread, which represents Oracle's `NLS_TIMESTAMP_FORMAT` parameter.

`ArgumentNullException` - The *`tsStr`* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Remarks

Example

```
// C#
// Set the nls_timestamp_format for the Parse() method
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStampLTZ from a string using the format specified.

OracleTimeStampLTZ ts = OracleTimeStamp.Parse("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString() method
og.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(ts.ToString()); // Prints "1999-NOV-11 11:02:33.444000000 AM"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

SetPrecision

This static method returns a new instance of an `OracleTimeStampLTZ` with the specified fractional second precision.

Declaration

```
// C#
public static OracleTimeStampLTZ SetPrecision(OracleTimeStampLTZ value1, int
fracSecPrecision);
```

Parameters

- *value1*

  The provided `OracleTimeStampLTZ` object.

- *fracSecPrecision*

  The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleTimeStampLTZ` structure with the specified fractional second precision

Exceptions

`ArgumentOutOfRangeException` - *fracSecPrecision* is out of the specified range.

Remarks

The value specified in the supplied *fracSecPrecision* parameter is used to perform a rounding off operation on the supplied `OracleTimeStampLTZ` value. Depending on this value, `0` or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleTimeStampLTZ` with a value of "`December 31, 9999 23:59:59.99`" results in the string "`December 31, 9999 23:59:59.99000`" when `SetPrecision()` is called with the fractional second precision set to `5`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ Static Type Operators

The `OracleTimeStampLTZ` static type operators are listed in Table 5–109.

Table 5–109 OracleTimeStampLTZ Static Operators

Operator	Description
operator+	Adds the supplied instance value to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure (Overloaded)
operator ==	Determines if two `OracleTimeStampLTZ` values are equal
operator >	Determines if the first of two `OracleTimeStampLTZ` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second
operator !=	Determines if two `OracleTimeStampLTZ` values are not equal
operator <	Determines if the first of two `OracleTimeStampLTZ` values is less than the second
operator <=	Determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator+

`operator+` adds the supplied value to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

Overload List:

- operator + (OracleTimeStampLTZ, OracleIntervalDS)

  This static operator adds the supplied `OracleIntervalDS` to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

- operator + (OracleTimeStampLTZ, OracleIntervalYM)

  This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

- operator + (OracleTimeStampLTZ, TimeSpan)

  This static operator adds the supplied `TimeSpan` to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator + (OracleTimeStampLTZ, OracleIntervalDS)

This static operator adds the supplied `OracleIntervalDS` to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static operator +(OracleTimeStampLTZ value1, OracleIntervalDS value2);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ`.

- *value2*

  An `OracleIntervalDS`.

Return Value

An `OracleTimeStampLTZ`.

Remarks

If either parameter has a null value, the returned `OracleTimeStampLTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator + (OracleTimeStampLTZ, OracleIntervalYM)

This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static operator +(OracleTimeStampLTZ value1,  OracleIntervalYM value2);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ`.

- *value2*

  An `OracleIntervalYM`.

Return Value

An `OracleTimeStampLTZ`.

Remarks

If either parameter has a null value, the returned `OracleTimeStampLTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator + (OracleTimeStampLTZ, TimeSpan)

This static operator adds the supplied `TimeSpan` to the supplied `OracleTimeStampLTZ` and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static operator +(OracleTimeStampLTZ value1, TimeSpan value2);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ`.

- *value2*

  A `TimeSpan`.

Return Value

An `OracleTimeStampLTZ`.

Remarks

If the `OracleTimeStampLTZ` instance has a null value, the returned `OracleTimeStampLTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator ==

This static operator determines if two `OracleTimeStampLTZ` values are equal.

Declaration

```
// C#
public static bool operator == (OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if they are the same; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator >

This static operator determines if the first of two `OracleTimeStampLTZ` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first `OracleTimeStampLTZ` value is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator >=

This static operator determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ`.

- *value2*

Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first `OracleTimeStampLTZ` is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator !=

This static operator determines if two `OracleTimeStampLTZ` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if two `OracleTimeStampLTZ` values are not equal; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator <

This static operator determines if the first of two `OracleTimeStampLTZ` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first `OracleTimeStampLTZ` is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator <=

This static operator determines if the first of two `OracleTimeStampLTZ` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleTimeStampLTZ value1, OracleTimeStampLTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampLTZ`.

- *value2*

  Second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first `OracleTimeStampLTZ` is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator -

`operator-` subtracts the supplied value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

Overload List:

- operator - (OracleTimeStampLTZ, OracleIntervalDS)

  This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampLTZ` value, and return a new `OracleTimeStampLTZ` structure.

- operator - (OracleTimeStampLTZ, OracleIntervalYM)

  This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

- operator - (OracleTimeStampLTZ, TimeSpan)

  This static operator subtracts the supplied `TimeSpan` value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator - (OracleTimeStampLTZ, OracleIntervalDS)

This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampLTZ` value, and return a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static operator - (OracleTimeStampLTZ value1, OracleIntervalDS value2);
```

Parameters

- *value1*

An `OracleTimeStampLTZ`.

- *`value2`*

  An `OracleIntervalDS` instance.

Return Value

An `OracleTimeStampLTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampLTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator - (OracleTimeStampLTZ, OracleIntervalYM)

This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static operator - (OracleTimeStampLTZ value1, OracleIntervalYM value2);
```

Parameters

- *`value1`*

  An `OracleTimeStampLTZ`.

- *`value2`*

  An `OracleIntervalYM`.

Return Value

An `OracleTimeStampLTZ` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampLTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator - (OracleTimeStampLTZ, TimeSpan)

This static operator subtracts the supplied `TimeSpan` value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static operator -(OracleTimeStampLTZ value1, TimeSpan value2);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ`.

- *value2*

  A `TimeSpan`.

Return Value

An `OracleTimeStampLTZ` structure.

Remarks

If the `OracleTimeStampLTZ` instance has a null value, the returned `OracleTimeStampLTZ` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ Static Type Conversions

The `OracleTimeStampLTZ` static type conversions are listed in Table 5–110.

Table 5–110 OracleTimeStampLTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampLTZ	Converts an instance value to an `OracleTimeStampLTZ` structure (Overloaded)
implicit operator OracleTimeStampLTZ	Converts an instance value to an `OracleTimeStampLTZ` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStampLTZ` value to a `DateTime` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ

`explicit operator OracleTimeStampLTZ` converts the supplied value to an `OracleTimeStampLTZ` structure.

Overload List:

- explicit operator OracleTimeStampLTZ(OracleTimeStamp)

  This static type conversion operator converts an `OracleTimeStamp` value to an `OracleTimeStampLTZ` structure.

- explicit operator OracleTimeStampLTZ(OracleTimeStampTZ)

  This static type conversion operator converts an `OracleTimeStampTZ` value to an `OracleTimeStampLTZ` structure.

- explicit operator OracleTimeStampLTZ(string)

  This static type conversion operator converts the supplied string to an `OracleTimeStampLTZ` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ(OracleTimeStamp)

This static type conversion operator converts an `OracleTimeStamp` value to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampLTZ (OracleTimeStamp value1);
```

Parameters

- *value1*

  An `OracleTimeStamp`.

Return Value

The `OracleTimeStampLTZ` structure contains the date and time of the `OracleTimeStampTZ` structure.

Remarks

If the `OracleTimeStamp` structure has a null value, the returned `OracleTimeStampLTZ` structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ(OracleTimeStampTZ)

This static type conversion operator converts an `OracleTimeStampTZ` value to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampLTZ (OracleTimeStampTZ value1);
```

Parameters

- *value1*

  An `OracleTimeStampTZ` instance.

Return Value

The `OracleTimeStampLTZ` structure contains the date and time in the `OracleTimeStampTZ` structure (which is normalized to the client local time zone).

Remarks

If the `OracleTimeStampTZ` structure has a null value, the returned `OracleTimeStampLTZ` structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ(string)

This static type conversion operator converts the supplied string to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampLTZ (string tsStr);
```

Parameters

- *tsStr*

  A string representation of an Oracle `TIMESTAMP WITH LOCAL TIME ZONE`.

Return Value

A `OracleTimeStampLTZ`.

Exceptions

ArgumentException - The*tsStr* parameter is an invalid string representation of an Oracle TIMESTAMP WITH LOCAL TIME ZONE or the *tsStr* is not in the timestamp format specified by the thread's OracleGlobalization.TimeStampFormat property, which represents Oracle's NLS_TIMESTAMP_FORMAT parameter.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_format for the OracleTimeStampLTZ(string) constructor
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStampLTZ from a string using the format specified.

OracleTimeStampLTZ ts=new OracleTimeStampLTZ("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString() method

og.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(ts.ToString()); // Prints "1999-NOV-11 11:02:33.444000000 AM"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39
- *Oracle Database SQL Reference* for further information on datetime format elements

implicit operator OracleTimeStampLTZ

`implicit operator OracleTimeStampLTZ` converts the supplied structure to an `OracleTimeStampLTZ` structure.

Overload List:

- implicit operator OracleTimeStampLTZ(OracleDate)

  This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampLTZ` structure.

- implicit operator OracleTimeStampLTZ(DateTime)

  This static type conversion operator converts a `DateTime` structure to an `OracleTimeStampLTZ` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

implicit operator OracleTimeStampLTZ(OracleDate)

This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public static implicit operator OracleTimeStampLTZ(OracleDate value1);
```

Parameters

- *value1*

  An OracleDate.

Return Value

The returned OracleTimeStampLTZ structure contains the date and time in the OracleDate structure.

Remarks

If the OracleDate structure has a null value, the returned OracleTimeStampLTZ structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

implicit operator OracleTimeStampLTZ(DateTime)

This static type conversion operator converts a DateTime structure to an OracleTimeStampLTZ structure.

Declaration

```
// C#
public static implicit operator OracleTimeStampLTZ(DateTime value1);
```

Parameters

- *value1*

  A DateTime structure.

Return Value

An OracleTimeStampLTZ structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator DateTime

This static type conversion operator converts an `OracleTimeStampLTZ` value to a `DateTime` structure.

Declaration

```
// C#
public static explicit operator DateTime(OracleTimeStampLTZ value1);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ` instance.

Return Value

A `DateTime` that contains the date and time in the current instance.

Exceptions

`OracleNullValueException` - The `OracleTimeStampLTZ` structure has a null value.

Remarks

The precision of the `OracleTimeStampLTZ` value can be lost during the conversion.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ Properties

The `OracleTimeStampLTZ` properties are listed in Table 5–111.

Table 5–111 OracleTimeStampLTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an `Oracle TIMESTAMP WITH LOCAL TIME ZONE` in Oracle internal format
Day	Specifies the `day` component of an `OracleTimeStampLTZ`
IsNull	Indicates whether the `OracleTimeStampLTZ` instance has a null value
Hour	Specifies the hour component of an `OracleTimeStampLTZ`
Millisecond	Specifies the millisecond component of an `OracleTimeStampLTZ`
Minute	Specifies the minute component of an `OracleTimeStampLTZ`
Month	Specifies the month component of an `OracleTimeStampLTZ`
Nanosecond	Specifies the nanosecond component of an `OracleTimeStampLTZ`
Second	Specifies the second component of an `OracleTimeStampLTZ`
Value	Specifies the date and time that is stored in the `OracleTimeStampLTZ` structure
Year	Specifies the year component of an `OracleTimeStampLTZ`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

BinData

This property returns an array of bytes that represents an Oracle `TIMESTAMP WITH LOCAL TIME ZONE` in Oracle internal format.

Declaration

```
// C#
public byte[] BinData {get;}
```

Property Value

A byte array that represents an Oracle `TIMESTAMP WITH LOCAL TIME ZONE` internal format.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Day

This property specifies the day component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Day{get;}
```

Property Value

A number that represents the day. Range of `Day` is (1 to 31).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

IsNull

This property indicates whether the current instance has a null value.

Declaration

```
// C#
public bool IsNull{get;}
```

Property Value

Returns `true` if the current instance contains a null value; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Hour

This property specifies the hour component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of `Hour` is (0 to 23).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Millisecond

This property gets the millisecond component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public double Millisecond{get;}
```

Property Value

A number that represents a millisecond. Range of `Millisecond` is (0 to 999.999999)

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Minute

This property gets the minute component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Minute{get;}
```

Property Value

A number that represent a minute. Range of `Minute` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Month

This property gets the month component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Month{get;}
```

Property Value

A number that represents a month. Range of `Month` is (1 to 12).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Nanosecond

This property gets the nanosecond component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Nanosecond{get;}
```

Property Value

A number that represents a nanosecond. Range of `Nanosecond` is (0 to 999999999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Second

This property gets the second component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Second{get;}
```

Property Value

A number that represents a second. Range of `Second` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Value

This property specifies the date and time that is stored in the `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public DateTime Value{get;}
```

Property Value

A `DateTime`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Year

This property gets the year component of an `OracleTimeStampLTZ`.

Declaration

```
// C#
public int Year{get;}
```

Property Value

A number that represents a year. The range of `Year` is (-4712 to 9999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ Methods

The `OracleTimeStampLTZ` methods are listed in Table 5–112.

Table 5–112 OracleTimeStampLTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStampLTZ` instance to an object and returns an integer that represents their relative values
Equals	Determines whether an object has the same date and time as the current `OracleTimeStampLTZ` instance (Overloaded)
GetHashCode	Returns a hash code for the `OracleTimeStampLTZ` instance
GetDaysBetween	Subtracts an `OracleTimeStampLTZ` from the current instance and returns an `OracleIntervalDS` that represents the difference

Table 5–112 OracleTimeStampLTZ Methods (Cont.)

Methods	Description
GetYearsBetween	Subtracts an `OracleTimeStampLTZ` from the current instance and returns an `OracleIntervalYM` that represents the difference
GetType	Inherited from `Object`
ToOracleDate	Converts the current `OracleTimeStampLTZ` structure to an `OracleDate` structure
ToOracleTimeStamp	Converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStamp` structure
ToOracleTimeStampTZ	Converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStampTZ` structure
ToString	Converts the current `OracleTimeStampLTZ` structure to a string
ToUniversalTime	Converts the current local time to Coordinated Universal Time (UTC)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddDays

This method adds the supplied number of days to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddDays(double days);
```

Parameters

- *days*

  The supplied number of days. Range is (-1,000,000,000 < *days* < 1,000,000,000)

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddHours(double hours);
```

Parameters

- *hours*

  The supplied number of hours. Range is (-24,000,000,000 < *hours* < 24,000,000,000).

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddMilliseconds(double milliseconds);
```

Parameters

- *milliseconds*

  The supplied number of milliseconds. Range is (-8.64 * 1016< `milliseconds` < 8.64 * 1016).

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddMinutes(double minutes);
```

Parameters

- *minutes*

  The supplied number of minutes. Range is (-1,440,000,000,000 < *minutes* < 1,440,000,000,000).

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddMonths

This method adds the supplied number of months to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddMonths(long months);
```

Parameters

- *months*

  The supplied number of months. Range is (-12,000,000,000 < *months* < 12,000,000,000).

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddNanoseconds(long nanoseconds);
```

Parameters

- *nanoseconds*

  The supplied number of nanoseconds.

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddSeconds

This method adds the supplied number of seconds to the current instance.

Declaration

```
// C#
public OracleTimeStampLTZ AddSeconds(double seconds);
```

Parameters

- *seconds*

  The supplied number of seconds. Range is (-8.64 * 1013< `seconds` < 8.64 * 1013).

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddYears

This method adds the supplied number of years to the current instance

Declaration

```
// C#
public OracleTimeStampLTZ AddYears(int years);
```

Parameters

- *years*

  The supplied number of years. Range is (-999,999,999 <= *years* < = 999,999,999)

Return Value

An `OracleTimeStampLTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

CompareTo

This method compares the current `OracleTimeStampLTZ` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  The object being compared to the current `OracleTimeStampLTZ` instance.

Return Value

The method returns a number that is:

- Less than zero: if the current `OracleTimeStampLTZ` instance value is less than that of *obj*.
- Zero: if the current `OracleTimeStampLTZ` instance and *obj* values are equal.
- Greater than zero: if the current `OracleTimeStampLTZ` instance value is greater than that of *obj*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *obj* parameter is not of type `OracleTimeStampLTZ`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleTimeStampLTZ`s. For example, comparing an `OracleTimeStampLTZ` instance with an `OracleBinary` instance is not allowed. When an `OracleTimeStampLTZ` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Equals

Overrides `Object`

This method determines whether an object has the same date and time as the current `OracleTimeStampLTZ` instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *obj*

  The object being compared to the current `OracleTimeStampLTZ` instance.

Return Value

Returns `true` if the *obj* is of type `OracleTimeStampLTZ` and represents the same date and time; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleTimeStampLTZ` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GetDaysBetween

This method subtracts an `OracleTimeStampLTZ` value from the current instance and returns an `OracleIntervalDS` that represents the difference.

Declaration

```
// C#
public OracleIntervalDS GetDaysBetween(OracleTimeStampLTZ value1);
```

Parameters

- *value1*

  The `OracleTimeStampLTZ` value being subtracted.

Return Value

An `OracleIntervalDS` that represents the interval between two `OracleTimeStampLTZ` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalDS` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GetYearsBetween

This method subtracts an `OracleTimeStampLTZ` value from the current instance and returns an `OracleIntervalYM` that represents the time interval.

Declaration

```
// C#
public OracleIntervalYM GetYearsBetween(OracleTimeStampLTZ value1);
```

Parameters

- *value1*

  The `OracleTimeStampLTZ` value being subtracted.

Return Value

An `OracleIntervalYM` that represents the interval between two `OracleTimeStampLTZ` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalYM` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

ToOracleDate

This method converts the current `OracleTimeStampLTZ` structure to an `OracleDate` structure.

Declaration

```
// C#
public OracleDate ToOracleDate();
```

Return Value

The returned `OracleDate` structure contains the date and time in the current instance.

Remarks

The precision of the `OracleTimeStampLTZ` value can be lost during the conversion.

If the current instance has a null value, the value of the returned `OracleDate` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

ToOracleTimeStamp

This method converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStamp` structure.

Declaration

```
// C#
public OracleTimeStamp ToOracleTimeStamp();
```

Return Value

The returned `OracleTimeStamp` contains the date and time in the current instance.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStamp` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

ToOracleTimeStampTZ

This method converts the current `OracleTimeStampLTZ` structure to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public OracleTimeStampTZ ToOracleTimeStampTZ();
```

Return Value

The returned `OracleTimeStampTZ` contains the date and time of the current instance, with the time zone set to the `OracleGlobalization.TimeZone` from the thread.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

ToString

Overrides `Object`

This method converts the current `OracleTimeStampLTZ` structure to a string.

Declaration

```
// C#
public override string ToString();
```

Return Value

A `string` that represents the same date and time as the current `OracleTimeStampLTZ` structure.

Remarks

The returned value is a string representation of the `OracleTimeStampLTZ` in the format specified by the `OracleGlobalization.TimeStampFormat` property of the thread.

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_format for the OracleTimeStampLTZ(string) constructor
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStampLTZ from a string using the format specified.

OracleTimeStampLTZ ts=new OracleTimeStampLTZ("11-NOV-1999 11:02:33.444 AM");

// Set the nls_timestamp_format for the ToString() method
og.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(ts.ToString()); // Prints "1999-NOV-11 11:02:33.444000000 AM"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

ToUniversalTime

This method converts the current local time to Coordinated Universal Time (UTC).

Declaration

```
// C#
public OracleTimeStampTZ ToUniversalTime();
```

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampTZ Structure

The `OracleTimeStampTZ` structure represents the Oracle `TIMESTAMP WITH TIME ZONE` data type to be stored in or retrieved from a database. Each `OracleTimeStampTZ` stores the following information: year, month, day, hour, minute, second, nanosecond, and time zone.

Class Inheritance

```
Object
  ValueType
    OracleTimeStampTZ
```

Declaration

```
// C#
public struct OracleTimeStampTZ : IComparable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

Example

```
// C#
// Illustrates usage of OracleTimeStampTZ

// Set the nls parameters for the current thread
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeZone = "US/Eastern";
og.TimeStampFormat   = "DD-MON-YYYY HH:MI:SS.FF AM";
og.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

// Create an OracleTimeStampTZ in US/Pacific time zone
OracleTimeStampTZ tstz1=new OracleTimeStampTZ("11-NOV-1999 "+
    "11:02:33.444 AM US/Pacific");

// Note that ToOracleTimeStampTZ uses the thread's time zone region,"US/Eastern"
OracleTimeStamp ts = new OracleTimeStamp("11-NOV-1999 11:02:33.444 AM");
```

```
OracleTimeStampTZ  tstz2 = ts.ToOracleTimeStampTZ();

// Calculate the difference between tstz1 and tstz2
OracleIntervalDS idsDiff = tstz1.GetDaysBetween(tstz2);

// Display information
Console.WriteLine("tstz1.TimeZone  = " + tstz1.TimeZone); //Prints "US/Pacific"
Console.WriteLine("tstz2.TimeZone  = " + tstz2.TimeZone); // Prints "US/Eastern"
Console.WriteLine("idsDiff.Hours   = " + idsDiff.Hours);  // Prints 3

Console.WriteLine("idsDiff.Minutes = " + idsDiff.Minutes); // Prints 0
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Members
- OracleTimeStampTZ Constructors
- OracleTimeStampTZ Static Fields
- OracleTimeStampTZ Static Methods
- OracleTimeStampTZ Static Operators
- OracleTimeStampTZ Static Type Conversions
- OracleTimeStampTZ Properties
- OracleTimeStampTZ Methods

OracleTimeStampTZ Members

`OracleTimeStampTZ` members are listed in the following tables:

OracleTimeStampTZ Constructors

`OracleTimeStampTZ` constructors are listed in Table 5–113

Table 5–113 OracleTimeStampTZ Constructors

Constructor	Description
OracleTimeStampTZ Constructors	Instantiates a new instance of `OracleTimeStampTZ` structure (Overloaded)

OracleTimeStampTZ Static Fields

The `OracleTimeStampTZ` static fields are listed in Table 5–114.

Table 5–114 OracleTimeStampTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStampTZ` structure in UTC, which is December 31, 999923:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStampTZ` structure in UTC, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStampTZ` structure

OracleTimeStampTZ Static Methods

The `OracleTimeStampTZ` static methods are listed in Table 5–115.

Table 5–115 OracleTimeStampTZ Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStampTZ` values are equal (Overloaded)
GetSysDate	Gets an `OracleTimeStampTZ` structure that represents the current date and time
GreaterThan	Determines if the first of two `OracleTimeStampTZ` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second

Table 5–115 OracleTimeStampTZ Static Methods (Cont.)

Methods	Description
LessThan	Determines if the first of two `OracleTimeStampTZ` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second
NotEquals	Determines if two `OracleTimeStampTZ` values are not equal
Parse	Gets an `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStampTZ` with the specified fractional second precision

OracleTimeStampTZ Static Operators

The `OracleTimeStampTZ` static operators are listed in Table 5–116.

Table 5–116 OracleTimeStampTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure (Overloaded)
operator ==	Determines if two `OracleTimeStampTZ` values are equal
operator >	Determines if the first of two `OracleTimeStampTZ` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second
operator !=	Determines if two `OracleTimeStampTZ` values are not equal
operator <	Determines if the first of two `OracleTimeStampTZ` values is less than the second

Table 5–116 OracleTimeStampTZ Static Operators (Cont.)

Operator	Description
operator <=	Determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure (Overloaded)

OracleTimeStampTZ Static Type Conversions

The `OracleTimeStampTZ` static type conversions are listed in Table 5–117.

Table 5–117 OracleTimeStampTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampTZ	Converts an instance value to an `OracleTimeStampTZ` structure (Overloaded)
implicit operator OracleTimeStampTZ	Converts an instance value to an `OracleTimeStampTZ` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStampTZ` value to a `DateTime` structure in the current time zone

OracleTimeStampTZ Properties

The `OracleTimeStampTZ` properties are listed in Table 5–118.

Table 5–118 OracleTimeStampTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle `TIMESTAMP WITH TIME ZONE` in Oracle internal format
Day	Specifies the day component of an `OracleTimeStampTZ` in the current time zone
IsNull	Indicates whether the current instance has a null value
Hour	Specifies the hour component of an `OracleTimeStampTZ` in the current time zone

Table 5–118 OracleTimeStampTZ Properties (Cont.)

Properties	Description
Millisecond	Specifies the millisecond component of an `OracleTimeStampTZ` in the current time zone
Minute	Specifies the minute component of an `OracleTimeStampTZ` in the current time zone
Month	Specifies the month component of an `OracleTimeStampTZ` in the current time zone
Nanosecond	Specifies the nanosecond component of an `OracleTimeStampTZ` in the current time zone
Second	Specifies the second component of an `OracleTimeStampTZ` in the current time zone
TimeZone	Returns the time zone of the `OracleTimeStampTZ` instance
Value	Returns the date and time that is stored in the `OracleTimeStampTZ` structure in the current time zone
Year	Specifies the year component of an `OracleTimeStampTZ`

OracleTimeStampTZ Methods

The `OracleTimeStampTZ` methods are listed in Table 5–119.

Table 5–119 OracleTimeStampTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance

Table 5–119 OracleTimeStampTZ Methods (Cont.)

Methods	Description
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance
CompareTo	Compares the current `OracleTimeStampTZ` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether an object has the same date and time as the current `OracleTimeStampTZ` instance
GetDaysBetween	Subtracts an `OracleTimeStampTZ` from the current instance and returns an `OracleIntervalDS` that represents the time interval
GetHashCode	Returns a hash code for the `OracleTimeStampTZ` instance
GetTimeZoneOffset	Gets the time zone information in hours and minutes of the current `OracleTimeStampTZ`
GetYearsBetween	Subtracts an `OracleTimeStampTZ` from the current instance and returns an `OracleIntervalYM` that represents the time interval
GetType	Inherited from `Object`
ToLocalTime	Converts the current `OracleTimeStampTZ` instance to local time
ToOracleDate	Converts the current `OracleTimeStampTZ` structure to an `OracleDate` structure
ToOracleTimeStampLTZ	Converts the current `OracleTimeStampTZ` structure to an `OracleTimeStampLTZ` structure
ToOracleTimeStamp	Converts the current `OracleTimeStampTZ` structure to an `OracleTimeStamp` structure
ToString	Converts the current `OracleTimeStampTZ` structure to a string

Table 5–119 OracleTimeStampTZ Methods (Cont.)

Methods	Description
ToUniversalTime	Converts the current datetime to Coordinated Universal Time (UTC)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure

OracleTimeStampTZ Constructors

The `OracleTimeStampTZ` constructors create new instances of the `OracleTimeStampTZ` structure.

Overload List:

- OracleTimeStampTZ(DateTime)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied `DateTime` value.

- OracleTimeStampTZ(DateTime, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied `DateTime` value and the supplied time zone data.

- OracleTimeStampTZ(string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string.

- OracleTimeStampTZ(int, int, int)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, and day.

- OracleTimeStampTZ(int, int, int, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, and time zone data.

- OracleTimeStampTZ(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.

- OracleTimeStampTZ(int, int, int, int, int, int, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and time zone data.

- OracleTimeStampTZ(int, int, int, int, int, int, double)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

- OracleTimeStampTZ(int, int, int, int, int, int, double, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, millisecond, and time zone data.

- OracleTimeStampTZ(int, int, int, int, int, int, int)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

- OracleTimeStampTZ(int, int, int, int, int, int, int, string)

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, nanosecond, and time zone data.

- OracleTimeStampTZ(byte [ ])

  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value to the provided byte array, that represents the internal Oracle `TIMESTAMP WITH TIME ZONE` format.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(DateTime)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied `DateTime` value.

Declaration

```
// C#
public OracleTimeStampTZ (DateTime dt);
```

Parameters

- *dt*

  The supplied `DateTime` value.

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

Exceptions

`ArgumentException` - The *dt* parameter cannot be used to construct a valid `OracleTimeStampTZ`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(DateTime, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure with the supplied `DateTime` value and the time zone data.

Declaration

```
// C#
public OracleTimeStampTZ (DateTime value1, string timeZone);
```

Parameters

- *value1*

  The supplied `DateTime` value.

- *timeZone*

The time zone data provided.

Exceptions

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ`.

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note: PST is a time zone region name as well as a time zone abbreviation, therefore it is accepted by `OracleTimeStampTZ`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#
public OracleTimeStampTZ (string tsStr);
```

Parameters

- *`tsStr`*

  A string that represents an Oracle `TIMESTAMP WITH TIME ZONE`.

Exceptions

`ArgumentException` - The *`tsStr`* is an invalid string representation of an Oracle `TIMESTAMP WITH TIME ZONE` or the *`tsStr`* is not in the timestamp format

specified by the `OracleGlobalization.TimeStampTZFormat` property of the thread.

`ArgumentNullException` - The *`tsStr`* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_tz_format OracleTimeStampTZ(string) constructor
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStampTZ from a string using the format specified.
OracleTimeStampTZ tstz = new OracleTimeStampTZ("11-NOV-1999" +
       "11:02:33.444 AM US/Pacific");

// Set the nls_timestamp_tz_format for the ToString() method
og.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(tstz.ToString());
// Prints "1999-NOV-11 11:02:33.444000000 AM  US/Pacific"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39
- *Oracle Database SQL Reference* for further information on date format elements

OracleTimeStampTZ(int, int, int)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, and day.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, string timeZone);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *timeZone*

  The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month or the time zone is invalid).

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note: PST is a time zone region name as well as a time zone abbreviation, therefore it is accepted by `OracleTimeStampTZ`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour, int minute,
  int second);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(int, int, int, int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ (int year, int month, int day, int hour, int minute,
 int second, string timeZone);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

The second provided. Range of *second* is (0 to 59).

- *timeZone*

  The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range of the month or the time zone is invalid).

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note: PST is a time zone region name as well as a time zone abbreviation, therefore it is accepted by `OracleTimeStampTZ`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(int, int, int, int, int, int, double)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour, int minute,
 int second, double millisecond);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

- *millisecond*

  The millisecond provided. Range of *millisecond* is (0 to 999.999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(int, int, int, int, int, int, double, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, millisecond, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour, int minute,
 int second, double millisecond, string timeZone);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

- *millisecond*

  The millisecond provided. Range of *millisecond* is (0 to 999.999999).

- *timeZone*

  The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month or the time zone is invalid).

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note: PST is a time zone region name as well as a time zone abbreviation, therefore it is accepted by `OracleTimeStampTZ`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(int, int, int, int, int, int, int)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour, int minute,
  int second, int nanosecond);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

- *nanosecond*

  The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month).

Remarks

The time zone is set to the `OracleGlobalization.TimeZone` of the thread.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(int, int, int, int, int, int, int, string)

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, nanosecond, and time zone data.

Declaration

```
// C#
public OracleTimeStampTZ(int year, int month, int day, int hour, int minute,
 int second, int nanosecond, string timeZone);
```

Parameters

- *year*

  The year provided. Range of *year* is (-4712 to 9999).

- *month*

  The month provided. Range of *month* is (1 to 12).

- *day*

  The day provided. Range of *day* is (1 to 31).

- *hour*

  The hour provided. Range of *hour* is (0 to 23).

- *minute*

  The minute provided. Range of *minute* is (0 to 59).

- *second*

  The second provided. Range of *second* is (0 to 59).

- *nanosecond*

  The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

- *timeZone*

  The time zone data provided.

Exceptions

`ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

`ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampTZ` (that is, the day is out of range for the month or the time zone is invalid).

Remarks

`timeZone` can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in `V$TIMEZONE_NAMES`, such as US/Pacific. Time zone abbreviations are not supported.

If time zone is null, the `OracleGlobalization.TimeZone` of the thread is used.

Note: PST is a time zone region name as well as a time zone abbreviation, therefore it is accepted by `OracleTimeStampTZ`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(byte [ ])

This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value to the provided byte array, that represents the internal Oracle `TIMESTAMP WITH TIME ZONE` format.

Declaration

```
// C#
public OracleTimeStampLTZ (byte[] bytes);
```

Parameters

- *bytes*

  The provided byte array that represents an Oracle `TIMESTAMP WITH TIME ZONE` in Oracle internal format.

Exceptions

`ArgumentException` - *bytes* is not in internal Oracle `TIMESTAMP WITH TIME ZONE` format or *bytes* is not a valid Oracle `TIMESTAMP WITH TIME ZONE`.

`ArgumentNullException` - *bytes* is null.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ Static Fields

The `OracleTimeStampTZ` static fields are listed in Table 5–120.

Table 5–120 OracleTimeStampTZ Static Fields

Field	Description
MaxValue	Represents the maximum valid date for an `OracleTimeStampTZ` structure in UTC, which is December 31, 999923:59:59.999999999
MinValue	Represents the minimum valid date for an `OracleTimeStampTZ` structure in UTC, which is January 1, -4712 0:0:0
Null	Represents a null value that can be assigned to an instance of the `OracleTimeStampTZ` structure

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

MaxValue

This static field represents the maximum valid datetime time for an `OracleTimeStampTZ` structure in UTC, which is December 31, 999923:59:59.999999999.

Declaration

```
// C#
public static readonly OracleTimeStampTZ MaxValue;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

MinValue

This static field represents the minimum valid datetime for an `OracleTimeStampTZ` structure in UTC, which is January 1, -4712 0:0:0.

Declaration

```
// C#
public static readonly OracleTimeStampTZ MinValue;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Null

This static field represents a null value that can be assigned to an instance of the `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static readonly OracleTimeStampTZ Null;
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ Static Methods

The `OracleTimeStampTZ` static methods are listed in Table 5–121.

Table 5–121 OracleTimeStampTZ Static Methods

Methods	Description
Equals	Determines if two `OracleTimeStampTZ` values are equal (Overloaded)
GetSysDate	Gets an `OracleTimeStampTZ` structure that represents the current date and time
GreaterThan	Determines if the first of two `OracleTimeStampTZ` values is greater than the second
GreaterThanOrEqual	Determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second
LessThan	Determines if the first of two `OracleTimeStampTZ` values is less than the second
LessThanOrEqual	Determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second

Table 5–121 OracleTimeStampTZ Static Methods (Cont.)

Methods	Description
NotEquals	Determines if two `OracleTimeStampTZ` values are not equal
Parse	Gets an `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string
SetPrecision	Returns a new instance of an `OracleTimeStampTZ` with the specified fractional second precision

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Equals

This static method determines if two `OracleTimeStampTZ` values are equal.

Declaration

```
// C#
public static bool Equals(OracleTimeStampTZ value1, OracleTimeStampTZ value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if two `OracleTimeStampTZ` values are equal. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.

- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

GetSysDate

This static method gets an `OracleTimeStampTZ` structure that represents the current date and time.

Declaration

```
// C#
public static OracleTimeStampTZ GetSysDate();
```

Return Value

An `OracleTimeStampTZ` structure that represents the current date and time.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

GreaterThan

This static method determines if the first of two `OracleTimeStampTZ` values is greater than the second.

Declaration

```
// C#
public static bool GreaterThan(OracleTimeStampTZ value1, OracleTimeStampTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

Second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampTZ` values is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

GreaterThanOrEqual

This static method determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second.

Declaration

```
// C#
public static bool GreaterThanOrEqual(OracleTimeStampTZ value1,
OracleTimeStampTZ value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampTZ` values is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

LessThan

This static method determines if the first of two `OracleTimeStampTZ` values is less than the second.

Declaration

```
// C#
public static bool LessThan(OracleTimeStampTZ value1, OracleTimeStampTZ value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampTZ` values is less than the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

LessThanOrEqual

This static method determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleTimeStampTZ value1,
  OracleTimeStampTZ value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first of two `OracleTimeStampTZ` values is less than or equal to the second. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

NotEquals

This static method determines if two `OracleTimeStampTZ` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleTimeStampTZ value1, OracleTimeStampTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if two `OracleTimeStampTZ` values are not equal. Returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Parse

This static method returns an `OracleTimeStampTZ` structure and sets its value for date and time using the supplied string.

Declaration

```
// C#
public static OracleTimeStampTZ Parse(string tsStr);
```

Parameters

- *tsStr*

  A string that represents an Oracle TIMESTAMP WITH TIME ZONE.

Return Value

An OracleTimeStampTZ structure.

Exceptions

ArgumentException - The *tsStr* is an invalid string representation of an Oracle TIMESTAMP WITH TIME ZONE or the *tsStr* is not in the timestamp format specified by the OracleGlobalization.TimeStampTZFormat property of the thread, which represents Oracle's NLS_TIMESTAMP_TZ_FORMAT parameter.

ArgumentNullException - The *tsStr* value is null.

Remarks

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's OracleGlobalization object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_tz_format for the Parse() method
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStampTZ from a string using the format specified.

OracleTimeStampTZ tstz = OracleTimeStampTZ.Parse("11-NOV-1999 " +
     "11:02:33.444 AM US/Pacific");

// Set the nls_timestamp_tz_format for the ToString() method
og.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(tstz.ToString());
// Prints "1999-NOV-11 11:02:33.444000000 AM  US/Pacific"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

SetPrecision

This static method returns a new instance of an `OracleTimeStampTZ` with the specified fractional second precision.

Declaration

```
// C#
public static OracleTimeStampTZ SetPrecision(OracleTimeStampTZ value1, int
fracSecPrecision);
```

Parameters

- *`value1`*

  The provided `OracleTimeStampTZ` object.

- *`fracSecPrecision`*

  The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value

An `OracleTimeStampTZ` structure with the specified fractional second precision

Exceptions

`ArgumentOutOfRangeException` - *`fracSecPrecision`* is out of the specified range.

Remarks

The value specified in the supplied *`fracSecPrecision`* is used to perform a rounding off operation on the supplied `OracleTimeStampTZ` value. Depending on this value, `0` or more trailing zeros are displayed in the string returned by `ToString()`.

Example

The `OracleTimeStampTZ` with a value of "`December 31, 9999 23:59:59.99 US/Pacific`" results in the string "`December 31, 9999 23:59:59.99000 US/Pacific`" when `SetPrecision()` is called with the fractional second precision set to `5`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ Static Operators

The `OracleTimeStampTZ` static operators are listed in Table 5–122.

Table 5–122 OracleTimeStampTZ Static Operators

Operator	Description
operator +	Adds the supplied instance value to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure (Overloaded)
operator ==	Determines if two `OracleTimeStampTZ` values are equal
operator >	Determines if the first of two `OracleTimeStampTZ` values is greater than the second
operator >=	Determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second
operator !=	Determines if two `OracleTimeStampTZ` values are not equal
operator <	Determines if the first of two `OracleTimeStampTZ` values is less than the second
operator <=	Determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second
operator -	Subtracts the supplied instance value from the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator +

`operator+` adds the supplied structure to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

Overload List:

- operator +(OracleTimeStampTZ, OracleIntervalDS)

  This static operator adds the supplied `OracleIntervalDS` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

- operator +(OracleTimeStampTZ, OracleIntervalYM)

  This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

- operator +(OracleTimeStampTZ, TimeSpan)

  This static operator adds the supplied `TimeSpan` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator +(OracleTimeStampTZ, OracleIntervalDS)

This static operator adds the supplied `OracleIntervalDS` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static operator +(OracleTimeStampTZ value1, OracleIntervalDS value2);
```

Parameters

- *value1*

  An `OracleTimeStampTZ`.

- *value2*

  An `OracleIntervalDS`.

Return Value

An `OracleTimeStampTZ`.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator +(OracleTimeStampTZ, OracleIntervalYM)

This static operator adds the supplied `OracleIntervalYM` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static operator +(OracleTimeStampTZ value1, OracleIntervalYM value2);
```

Parameters

- *value1*

  An `OracleTimeStampTZ`.

- *value2*

  An `OracleIntervalYM`.

Return Value

An `OracleTimeStampTZ`.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator +(OracleTimeStampTZ, TimeSpan)

This static operator adds the supplied `TimeSpan` to the supplied `OracleTimeStampTZ` and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static operator +(OracleTimeStampTZ value1,  TimeSpan value2);
```

Parameters

- *value1*

  An `OracleTimeStampTZ`.

- *value2*

  A `TimeSpan`.

Return Value

An `OracleTimeStampTZ`.

Remarks

If the `OracleTimeStampTZ` instance has a null value, the returned `OracleTimeStampTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator ==

This static operator determines if two `OracleTimeStampTZ` values are equal.

Declaration

```
// C#
public static bool operator == (OracleTimeStampTZ value1, OracleTimeStampTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if they are equal; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator >

This static operator determines if the first of two `OracleTimeStampTZ` values is greater than the second.

Declaration

```
// C#
public static bool operator > (OracleTimeStampTZ value1, OracleTimeStampTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first `OracleTimeStampTZ` value is greater than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator >=

This static operator determines if the first of two `OracleTimeStampTZ` values is greater than or equal to the second.

Declaration

```
// C#
public static bool operator >= (OracleTimeStampTZ value1, OracleTimeStampTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

Second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first `OracleTimeStampTZ` is greater than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator !=

This static operator determines if two `OracleTimeStampTZ` values are not equal.

Declaration

```
// C#
public static bool operator != (OracleTimeStampTZ value1, OracleTimeStampTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if two `OracleTimeStampTZ` values are not equal; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator <

This static operator determines if the first of two `OracleTimeStampTZ` values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleTimeStampTZ value1, OracleTimeStampTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first `OracleTimeStampTZ` is less than the second; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator <=

This static operator determines if the first of two `OracleTimeStampTZ` values is less than or equal to the second.

Declaration

```
// C#
public static bool operator <= (OracleTimeStampTZ value1, OracleTimeStampTZ
value2);
```

Parameters

- *value1*

  First `OracleTimeStampTZ`.

- *value2*

  Second `OracleTimeStampTZ`.

Return Value

Returns `true` if the first `OracleTimeStampTZ` is less than or equal to the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator -

`operator-` subtracts the supplied value, from the supplied `OracleTimeStampTZ` value, and returns a new `OracleTimeStampTZ` structure.

Overload List:

- operator - (OracleTimeStampTZ, OracleIntervalDS)

  This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampTZ` value, and return a new `OracleTimeStampTZ` structure.

- operator - (OracleTimeStampTZ, OracleIntervalYM)

  This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStampTZ` value, and returns a new `OracleTimeStampTZ` structure.

- operator - (OracleTimeStampTZ value1, TimeSpan value2)

  This static operator subtracts the supplied `TimeSpan` value, from the supplied `OracleTimeStampTZ` value, and returns a new `OracleTimeStampTZ` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator - (OracleTimeStampTZ, OracleIntervalDS)

This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampTZ` value, and return a new `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static operator - (OracleTimeStampTZ value1, OracleIntervalDS value2);
```

Parameters

- *value1*

  An `OracleTimeStampTZ`.

- *value2*

  An OracleIntervalDS.

Return Value

An OracleTimeStampTZ structure.

Remarks

If either parameter has a null value, the returned OracleTimeStampTZ has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator - (OracleTimeStampTZ, OracleIntervalYM)

This static operator subtracts the supplied OracleIntervalYM value, from the supplied OracleTimeStampTZ value, and returns a new OracleTimeStampTZ structure.

Declaration

```
// C#
public static operator - (OracleTimeStampTZ value1, OracleIntervalYM value2);
```

Parameters

- *value1*

  An OracleTimeStampTZ.

- *value2*

  An OracleIntervalYM.

Return Value

An OracleTimeStampTZ structure.

Remarks

If either parameter has a null value, the returned OracleTimeStampTZ has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator - (OracleTimeStampTZ value1, TimeSpan value2)

This static operator subtracts the supplied `TimeSpan` value, from the supplied `OracleTimeStampTZ` value, and returns a new `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static operator - (OracleTimeStampTZ value1, TimeSpan value2);
```

Parameters

- *value1*

  An `OracleTimeStampTZ`.

- *value2*

  A `TimeSpan`.

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If the `OracleTimeStampTZ` instance has a null value, the returned `OracleTimeStampTZ` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ Static Type Conversions

The `OracleTimeStampTZ` static type conversions are listed in Table 5–123.

Table 5–123 OracleTimeStampTZ Static Type Conversions

Operator	Description
explicit operator OracleTimeStampTZ	Converts an instance value to an `OracleTimeStampTZ` structure (Overloaded)
implicit operator OracleTimeStampTZ	Converts an instance value to an `OracleTimeStampTZ` structure (Overloaded)
explicit operator DateTime	Converts an `OracleTimeStampTZ` value to a `DateTime` structure in the current time zone

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

explicit operator OracleTimeStampTZ

`explicit operator OracleTimeStampTZ` converts an instance value to an `OracleTimeStampTZ` structure.

Overload List:

- explicit operator OracleTimeStampTZ(OracleTimeStamp)

  This static type conversion operator converts an `OracleTimeStamp` value to an `OracleTimeStampTZ` structure.

- explicit operator OracleTimeStampTZ(OracleTimeStampLTZ)

  This static type conversion operator converts an `OracleTimeStampLTZ` value to an `OracleTimeStampTZ` structure.

- explicit operator OracleTimeStampTZ(string)

  This static type conversion operator converts the supplied string value to an `OracleTimeStampTZ` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

explicit operator OracleTimeStampTZ(OracleTimeStamp)

This static type conversion operator converts an `OracleTimeStamp` value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampTZ(OracleTimeStamp value1);
```

Parameters

- *value1*

  An `OracleTimeStamp`.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleTimeStamp` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

The `OracleGlobalization.TimeZone` of the thread is used to convert from an `OracleTimeStamp` structure to an `OracleTimeStampTZ` structure.

If the `OracleTimeStamp` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

explicit operator OracleTimeStampTZ(OracleTimeStampLTZ)

This static type conversion operator converts an `OracleTimeStampLTZ` value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampTZ(OracleTimeStampLTZ value1);
```

Parameters

- *value1*

  An `OracleTimeStampLTZ`.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleTimeStampLTZ` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

If the `OracleTimeStampLTZ` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

explicit operator OracleTimeStampTZ(string)

This static type conversion operator converts the supplied string value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static explicit operator OracleTimeStampTZ(string tsStr);
```

Parameters

- *tsStr*

  A string representation of an Oracle `TIMESTAMP WITH TIME ZONE`.

Return Value

A `OracleTimeStampTZ` value.

Exceptions

`ArgumentException` - The *tsStr* is an invalid string representation of an Oracle `TIMESTAMP WITH TIME ZONE`. or the *tsStr* is not in the timestamp format specified by the thread's `OracleGlobalization.TimeStampTZFormat` property, which represents Oracle's `NLS_TIMESTAMP_TZ_FORMAT` parameter.

Remarks

The names and abbreviations used for months and days are in the language specified by the `DateLanguage` and `Calendar` properties of the thread's `OracleGlobalization` object. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_tz_format for the explicit operator
// OracleTimeStampTZ(string)
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStampTZ from a string using the format specified.
OracleTimeStampTZ tstz = new OracleTimeStampTZ("11-NOV-1999" +
       "11:02:33.444 AM US/Pacific");

// Set the nls_timestamp_tz_format for the ToString() method
```

```
og.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(tstz.ToString());
// Prints "1999-NOV-11 11:02:33.444000000 AM  US/Pacific"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

implicit operator OracleTimeStampTZ

`implicit operator OracleTimeStampTZ` converts a `DateTime` structure to an `OracleTimeStampTZ` structure.

Overload List:

- implicit operator OracleTimeStampTZ(OracleDate)

  This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampTZ` structure.

- implicit operator OracleTimeStampTZ(DateTime)

  This static type conversion operator converts a `DateTime` structure to an `OracleTimeStampTZ` structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

implicit operator OracleTimeStampTZ(OracleDate)

This static type conversion operator converts an `OracleDate` value to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static implicit operator OracleTimeStampTZ(OracleDate value1);
```

Parameters

- *value1*

  An `OracleDate`.

Return Value

The returned `OracleTimeStampTZ` contains the date and time from the `OracleDate` and the time zone from the `OracleGlobalization.TimeZone` of the thread.

Remarks

The `OracleGlobalization.TimeZone` of the thread is used to convert from an `OracleDate` to an `OracleTimeStampTZ` structure. If the `OracleDate` structure has a null value, the returned `OracleTimeStampTZ` structure also has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

implicit operator OracleTimeStampTZ(DateTime)

This static type conversion operator converts a `DateTime` structure to an `OracleTimeStampTZ` structure.

Declaration

```
// C#
public static implicit operator OracleTimeStampTZ (DateTime value1);
```

Parameters

- *value1*

  A DateTime structure.

Return Value

The returned OracleTimeStampTZ contains the date and time from the DateTime and the time zone from the OracleGlobalization.TimeZone of the thread.

Remarks

The OracleGlobalization.TimeZone of the thread is used to convert from a DateTime to an Oracle TimeStampTZ structure.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

explicit operator DateTime

This static type conversion operator converts an OracleTimeStampTZ value to a DateTime structure in the current time zone.

Declaration

```
// C#
public static explicit operator DateTime(OracleTimeStampTZ value1);
```

Parameters

- *value1*

  An OracleTimeStampTZ.

Return Value

A DateTime containing the date and time in the current instance, but with the time zone information in the current instance truncated.

Exceptions

`OracleNullValueException` - The `OracleTimeStampTZ` structure has a null value.

Remarks

The precision of the `OracleTimeStampTZ` value can be lost during the conversion, and the time zone information in the current instance is truncated

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ Properties

The `OracleTimeStampTZ` properties are listed in Table 5–124.

Table 5–124 OracleTimeStampTZ Properties

Properties	Description
BinData	Returns an array of bytes that represents an Oracle `TIMESTAMP WITH TIME ZONE` in Oracle internal format
Day	Specifies the day component of an `OracleTimeStampTZ` in the current time zone
IsNull	Indicates whether the current instance has a null value
Hour	Specifies the hour component of an `OracleTimeStampTZ` in the current time zone
Millisecond	Specifies the millisecond component of an `OracleTimeStampTZ` in the current time zone
Minute	Specifies the minute component of an `OracleTimeStampTZ` in the current time zone
Month	Specifies the month component of an `OracleTimeStampTZ` in the current time zone
Nanosecond	Specifies the nanosecond component of an `OracleTimeStampTZ` in the current time zone
Second	Specifies the second component of an `OracleTimeStampTZ` in the current time zone

Table 5–124 OracleTimeStampTZ Properties (Cont.)

Properties	Description
TimeZone	Returns the time zone of the `OracleTimeStampTZ` instance
Value	Returns the date and time that is stored in the `OracleTimeStampTZ` structure in the current time zone
Year	Specifies the year component of an `OracleTimeStampTZ`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

BinData

This property returns an array of bytes that represents an Oracle `TIMESTAMP WITH TIME ZONE` in Oracle internal format.

Declaration

```
// C#
public byte[] BinData {get;}
```

Property Value

The provided byte array that represents an Oracle `TIMESTAMP WITH TIME ZONE` in Oracle internal format.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Day

This property specifies the day component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public int Day{get;}
```

Property Value

A number that represents the day. Range of `Day` is (1 to 31).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

IsNull

This property indicates whether the current instance has a null value.

Declaration

```
// C#
public bool IsNull{get;}
```

Property Value

Returns `true` if the current instance has a null value. Otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Hour

This property specifies the hour component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public int Hour{get;}
```

Property Value

A number that represents the hour. Range of `Hour` is (0 to 23).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Millisecond

This property gets the millisecond component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public double Millisecond{get;}
```

Property Value

A number that represents a millisecond. Range of `Millisecond` is (0 to 999.999999)

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Minute

This property gets the minute component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
```

```
public int Minute{get;}
```

Property Value

A number that represent a minute. Range of `Minute` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Month

This property gets the month component of an `OracleTimeStampTZ` in the current time zone

Declaration

```
// C#
public int Month{get;}
```

Property Value

A number that represents a month. Range of `Month` is (1 to 12).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Nanosecond

This property gets the nanosecond component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public int Nanosecond{get;}
```

Property Value

A number that represents a nanosecond. Range of `Nanosecond` is (0 to 999999999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Second

This property gets the second component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public int Second{get;}
```

Property Value

A number that represents a second. Range of `Second` is (0 to 59).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

TimeZone

This property returns the time zone of the `OracleTimeStampTZ` instance.

Declaration

```
// C#
public string TimeZone{get;}
```

Property Value

A string that represents the time zone.

Remarks

If no time zone is specified in the constructor, this property is set to the thread's `OracleGlobalization.TimeZone` by default

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

Value

This property returns the date and time that is stored in the `OracleTimeStampTZ` structure in the current time zone.

Declaration

```
// C#
public DateTime Value{get;}
```

Property Value

A `DateTime` in the current time zone.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Year

This property sets the year component of an `OracleTimeStampTZ` in the current time zone.

Declaration

```
// C#
public int Year{get;}
```

Property Value

A number that represents a year. The range of `Year` is (-4712 to 9999).

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ Methods

The `OracleTimeStampTZ` methods are listed in Table 5–125.

Table 5–125 OracleTimeStampTZ Methods

Methods	Description
AddDays	Adds the supplied number of days to the current instance
AddHours	Adds the supplied number of hours to the current instance
AddMilliseconds	Adds the supplied number of milliseconds to the current instance
AddMinutes	Adds the supplied number of minutes to the current instance
AddMonths	Adds the supplied number of months to the current instance
AddNanoseconds	Adds the supplied number of nanoseconds to the current instance
AddSeconds	Adds the supplied number of seconds to the current instance
AddYears	Adds the supplied number of years to the current instance

Table 5–125 OracleTimeStampTZ Methods (Cont.)

Methods	Description
CompareTo	Compares the current `OracleTimeStampTZ` instance to an object, and returns an integer that represents their relative values
Equals	Determines whether an object has the same date and time as the current `OracleTimeStampTZ` instance (Overloaded)
GetDaysBetween	Subtracts an `OracleTimeStampTZ` from the current instance and returns an `OracleIntervalDS` that represents the time interval
GetHashCode	Returns a hash code for the `OracleTimeStampTZ` instance
GetTimeZoneOffset	Gets the time zone information in hours and minutes of the current `OracleTimeStampTZ`
GetYearsBetween	Subtracts an `OracleTimeStampTZ` from the current instance and returns an `OracleIntervalYM` that represents the time interval
GetType	Inherited from `Object`
ToLocalTime	Converts the current `OracleTimeStampTZ` instance to local time
ToOracleDate	Converts the current `OracleTimeStampTZ` structure to an `OracleDate` structure
ToOracleTimeStampLTZ	Converts the current `OracleTimeStampTZ` structure to an `OracleTimeStampLTZ` structure
ToOracleTimeStamp	Converts the current `OracleTimeStampTZ` structure to an `OracleTimeStamp` structure
ToString	Converts the current `OracleTimeStampTZ` structure to a string
ToUniversalTime	Converts the current datetime to Coordinated Universal Time (UTC)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddDays

This method adds the supplied number of days to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddDays(double days);
```

Parameters

- *days*

  The supplied number of days. Range is (-1,000,000,000 < *days* < 1,000,000,000)

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddHours(double hours);
```

Parameters

- *hours*

  The supplied number of hours. Range is (-24,000,000,000 < *hours* < 24,000,000,000).

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddMilliseconds(double milliseconds);
```

Parameters

- *milliseconds*

  The supplied number of milliseconds. Range is (-8.64 * 1016< `milliseconds` < 8.64 * 1016).

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddMinutes(double minutes);
```

Parameters

- *minutes*

  The supplied number of minutes. Range is (-1,440,000,000,000 < *minutes* < 1,440,000,000,000).

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddMonths

This method adds the supplied number of months to the current instance.

Declaration

```
// C#
```

```
public OracleTimeStampTZ AddMonths(long months);
```

Parameters

- *months*

  The supplied number of months. Range is (-12,000,000,000 < *months* < 12,000,000,000).

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddNanoseconds(long nanoseconds);
```

Parameters

- *nanoseconds*

  The supplied number of nanoseconds.

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddSeconds

This method adds the supplied number of seconds to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddSeconds(double seconds);
```

Parameters

- *seconds*

  The supplied number of seconds. Range is (-8.64 * 1013< `seconds` < 8.64 * 1013).

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddYears

This method adds the supplied number of years to the current instance

Declaration

```
// C#
```

```
public OracleTimeStampTZ AddYears(int years);
```

Parameters

- *years*

  The supplied number of years. Range is (-999,999,999 <= *years* < = 999,999,999).

Return Value

An `OracleTimeStampTZ`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

`ArgumentOutofRangeException` - The argument value is out of the specified range.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

CompareTo

This method compares the current `OracleTimeStampTZ` instance to an object, and returns an integer that represents their relative values.

Declaration

```
// C#
public int CompareTo(object obj);
```

Parameters

- *obj*

  The object being compared to the current `OracleTimeStampTZ` instance.

Return Value

The method returns a number that is:

Less than zero: if the current `OracleTimeStampTZ` instance value is less than that of *obj*.

Zero: if the current `OracleTimeStampTZ` instance and *`obj`* values are equal.

Greater than zero: if the current `OracleTimeStampTZ` instance value is greater than that of *`obj`*.

Implements

`IComparable`

Exceptions

`ArgumentException` - The *`obj`* is not of type `OracleTimeStampTZ`.

Remarks

The following rules apply to the behavior of this method.

- The comparison must be between `OracleTimeStampTZ`s. For example, comparing an `OracleTimeStampTZ` instance with an `OracleBinary` instance is not allowed. When an `OracleTimeStampTZ` is compared with a different type, an `ArgumentException` is thrown.
- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Equals

Overrides `Object`

This method determines whether an object has the same date and time as the current `OracleTimeStampTZ` instance.

Declaration

```
// C#
public override bool Equals(object obj);
```

Parameters

- *obj*

  The object being compared to the current OracleTimeStampTZ instance.

Return Value

Returns true if the *obj* is of type OracleTimeStampTZ and represents the same date and time; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampTZ that has a value is greater than an OracleTimeStampTZ that has a null value.
- Two OracleTimeStampTZs that contain a null value are equal.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

GetDaysBetween

This method subtracts an OracleTimeStampTZ value from the current instance and returns an OracleIntervalDS that represents the time interval.

Declaration

```
// C#
public OracleIntervalDS GetDaysBetween(OracleTimeStampTZ value1);
```

Parameters

- *value1*

  The OracleTimeStampTZ value being subtracted.

Return Value

An OracleIntervalDS that represents the interval between two OracleTimeStampTZ values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalDS` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

GetHashCode

Overrides `Object`

This method returns a hash code for the `OracleTimeStampTZ` instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

A number that represents the hash code.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

GetTimeZoneOffset

This method gets the time zone portion in hours and minutes of the current `OracleTimeStampTZ`.

Declaration

```
// C#
public TimeSpan GetTimeZoneOffset();
```

Return Value

A `TimeSpan`.

Exceptions

`OracleNullValueException` - The current instance has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

GetYearsBetween

This method subtracts an `OracleTimeStampTZ` value from the current instance and returns an `OracleIntervalYM` that represents the time interval.

Declaration

```
// C#
public OracleIntervalYM GetYearsBetween(OracleTimeStampTZ val);
```

Parameters

- *val*

  The `OracleTimeStampTZ` value being subtracted.

Return Value

An `OracleIntervalYM` that represents the interval between two `OracleTimeStampTZ` values.

Remarks

If either the current instance or the parameter has a null value, the returned `OracleIntervalYM` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

ToLocalTime

This method converts the current `OracleTimeStampTZ` instance to local time.

Declaration

```
// C#
public OracleTimeStampLTZ ToLocalTime();
```

Return Value

An `OracleTimeStampLTZ` that contains the date and time, which is normalized to the client local time zone, in the current instance.

Remarks

If the current instance has a null value, the returned `OracleTimeStampLTZ` has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

ToOracleDate

This method converts the current `OracleTimeStampTZ` structure to an `OracleDate` structure.

Declaration

```
// C#
public OracleDate ToOracleDate();
```

Return Value

The returned `OracleDate` contains the date and time in the current instance, but the time zone information in the current instance is truncated

Remarks

The precision of the `OracleTimeStampTZ` value can be lost during the conversion, and the time zone information in the current instance is truncated.

If the current instance has a null value, the value of the returned `OracleDate` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

ToOracleTimeStampLTZ

This method converts the current `OracleTimeStampTZ` structure to an `OracleTimeStampLTZ` structure.

Declaration

```
// C#
public OracleTimeStampLTZ ToOracleTimeStampLTZ();
```

Return Value

The returned `OracleTimeStampLTZ` structure contains the date and time, which is normalized to the client local time zone, in the current instance.

Remarks

If the value of the current instance has a null value, the value of the returned `OracleTimeStampLTZ` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

ToOracleTimeStamp

This method converts the current `OracleTimeStampTZ` structure to an `OracleTimeStamp` structure.

Declaration

```
// C#
public OracleTimeStamp ToOracleTimeStamp();
```

Return Value

The returned `OracleTimeStamp` contains the date and time in the current instance, but the time zone information is truncated.

Remarks

If the value of the current instance has a null value, the value of the returned `OracleTimeStamp` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

ToString

Overrides `Object`

This method converts the current `OracleTimeStampTZ` structure to a string.

Declaration

```
// C#
public override string ToString();
```

Return Value

A `string` that represents the same date and time as the current `OracleTimeStampTZ` structure.

Remarks

The returned value is a string representation of an `OracleTimeStampTZ` in the format specified by the `OracleGlobalization.TimeStampTZFormat` property of the thread. The names and abbreviations used for months and days are in the language specified by the `OracleGlobalization.DateLanguage` and the `OracleGlobalization.Calendar` properties of the thread. If any of the thread's globalization properties are set to null or an empty string, the client computer's settings are used.

Example

```
// C#
// Set the nls_timestamp_tz_format for the TimeStampTZFormat(string) constructor
```

```
OracleGlobalization og = OracleGlobalization.GetClientInfo();
og.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

// construct OracleTimeStampTZ from a string using the format specified.
OracleTimeStampTZ tstz = new OracleTimeStampTZ("11-NOV-1999" +
      "11:02:33.444 AM US/Pacific");

// Set the nls_timestamp_tz_format for the ToString() method
og.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(og);

Console.WriteLine(tstz.ToString());
// Prints "1999-NOV-11 11:02:33.444000000 AM  US/Pacific"
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 4-212
- "Globalization Support" on page 3-39

ToUniversalTime

This method converts the current datetime to Coordinated Universal Time (UTC).

Declaration

```
// C#
public OracleTimeStampTZ ToUniversalTime();
```

Return Value

An `OracleTimeStampTZ` structure.

Remarks

If the current instance has a null value, the value of the returned `OracleTimeStampTZ` structure has a null value.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

ODP.NET Type Exceptions

This section covers the ODP.NET Type exceptions.

- OracleTypeException Class
- OracleNullValueException Class
- OracleTruncateException Class

OracleTypeException Class

The `OracleTypeException` is the base exception class for handling exceptions that occur in the ODP.NET Types classes.

Class Inheritance

```
Object
  Exception
    SystemException
      OracleTypeException
```

Declaration

```
// C#
public class OracleTypeException : SystemException
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Members
- OracleTypeException Constructors
- OracleTypeException Static Methods
- OracleTypeException Properties
- OracleTypeException Methods

OracleTypeException Members

`OracleTypeException` members are listed in the following tables:

OracleTypeException Constructors

The `OracleTypeException` constructors are listed in Table 5–126.

Table 5–126 OracleTypeException Constructor

Constructor	Description
OracleTypeException Constructors	Creates a new instance of the `OracleTypeException` class (Overloaded)

OracleTypeException Static Methods

The `OracleTypeException` static methods are listed in Table 5–127.

Table 5–127 OracleTypeException Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleTypeException Properties

The `OracleTypeException` properties are listed in Table 5–128.

Table 5–128 OracleTypeException Properties

Properties	Description
`HelpLink`	Inherited from `Exception`
`InnerException`	Inherited from `Exception`
Message	Specifies the error messages that occur in the exception
Source	Specifies the name of the data provider that generates the error
`StackTrace`	Inherited from `Exception`
`TargetSite`	Inherited from `Exception`

OracleTypeException Methods

The `OracleTypeException` methods are listed in Table 5–129.

Table 5–129 OracleTypeException Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetBaseException`	Inherited from `Exception`
`GetHashCode`	Inherited from `Object`
`GetObjectData`	Inherited from `Exception`
`GetType`	Inherited from `Object`
ToString	Returns the fully qualified name of this exception

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class

OracleTypeException Constructors

The `OracleTypeException` constructors create new instances of the `OracleTypeException` class.

Overload List:

- OracleTypeException(string)

  This constructor creates a new instance of the `OracleTypeException` class with the specified error message, `errMessage`.

- OracleTypeException(SerializationInfo, StreamContext)

  This constructor creates a new instance of the `OracleTypeException` class with the specified serialization information, `si`, and the specified streaming context, `sc`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

OracleTypeException(string)

This constructor creates a new instance of the `OracleTypeException` class with the specified error message, `errMessage`.

Declaration

```
// C#
public OracleTypeException (string errMessage);
```

Parameters

- *errMessage*

  The specified error message.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

OracleTypeException(SerializationInfo, StreamContext)

This constructor creates a new instance of the `OracleTypeException` class with the specified serialization information, `si`, and the specified streaming context, `sc`.

Declaration

```
// C#
public OracleTypeException (SerializationInfo si, StreamingContext sc);
```

Parameters

- *si*

  The specified serialization information.

- *sc*

  The specified streaming context.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

OracleTypeException Static Methods

The `OracleTypeException` static methods are listed in Table 5–130.

Table 5–130 OracleTypeException Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

OracleTypeException Properties

The `OracleTypeException` properties are listed in Table 5–131.

Table 5–131 OracleTypeException Properties

Properties	Description
`HelpLink`	Inherited from `Exception`
`InnerException`	Inherited from `Exception`
Message	Specifies the error messages that occur in the exception
Source	Specifies the name of the data provider that generates the error
`StackTrace`	Inherited from `Exception`
`TargetSite`	Inherited from `Exception`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

Message

Overrides `Exception`

This property specifies the error messages that occur in the exception.

Declaration

```
// C#
public override string Message {get;}
```

Property Value

An error message.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

Source

Overrides `Exception`

This property specifies the name of the data provider that generates the error.

Declaration

```
// C#
public override string Source {get;}
```

Property Value

Oracle Data Provider for .NET.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

OracleTypeException Methods

The `OracleTypeException` methods are listed in Table 5–132.

Table 5–132 OracleTypeException Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetBaseException`	Inherited from `Exception`
`GetHashCode`	Inherited from `Object`
`GetObjectData`	Inherited from `Exception`
`GetType`	Inherited from `Object`
ToString	Returns the fully qualified name of this exception

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

ToString

Overrides `Exception`

This method returns the fully qualified name of this exception, the error message in the Message property, the `InnerException.ToString()` message, and the stack trace.

Declaration

```
// C#
public override string ToString();
```

Return Value

The fully qualified name of this exception.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTypeException Class
- OracleTypeException Members

OracleNullValueException Class

The `OracleNullValueException` represents an exception that is thrown when trying to access an ODP.NET Type structure that has a null value.

Class Inheritance

```
Object
  Exception
    SystemException
      OracleTypeException
        OracleNullValueException
```

Declaration

```
// C#
public sealed class OracleNullValueException : OracleTypeException
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleNullValueException Members
- OracleNullValueException Constructors
- OracleNullValueException Static Methods
- OracleNullValueException Properties
- OracleNullValueException Methods

OracleNullValueException Members

`OracleNullValueException` members are listed in the following tables:

OracleNullValueException Constructors

The `OracleNullValueException` constructors are listed in Table 5–133.

Table 5–133 OracleNullValueException Constructors

Constructor	Description
OracleNullValueException Constructors	Creates a new instance of the `OracleNullValueException` class (Overloaded)

OracleNullValueException Static Methods

The `OracleNullValueException` static methods are listed in Table 5–134.

Table 5–134 OracleNullValueException Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleNullValueException Properties

The `OracleNullValueException` properties are listed in Table 5–135.

Table 5–135 OracleNullValueException Properties

Properties	Description
`HelpLink`	Inherited from `Exception`
`InnerException`	Inherited from `Exception`
`Message`	Inherited from `OracleTypeException`
`Source`	Inherited from `OracleTypeException`
`StackTrace`	Inherited from `Exception`
`TargetSite`	Inherited from `Exception`

OracleNullValueException Methods

The `OracleNullValueException` methods are listed in Table 5–136.

Table 5–136 OracleNullValueException Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetBaseException`	Inherited from `Exception`
`GetHashCode`	Inherited from `Object`
`GetObjectData`	Inherited from `Exception`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `OracleTypeException`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleNullValueException Class

OracleNullValueException Constructors

The `OracleNullValueException` constructors create new instances of the `OracleNullValueException` class.

Overload List:

- OracleNullValueException()

  This constructor creates a new instance of the `OracleNullValueException` class with its default properties.

- OracleNullValueException(string)

  This constructor creates a new instance of the `OracleNullValueException` class with the specified error message, `errMessage`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleNullValueException Class
- OracleNullValueException Members

OracleNullValueException()

This constructor creates a new instance of the `OracleNullValueException` class with its default properties.

Declaration

```
// C#
public OracleNullValueException();
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleNullValueException Class
- OracleNullValueException Members

OracleNullValueException(string)

This constructor creates a new instance of the `OracleNullValueException` class with the specified error message, `errMessage`.

Declaration

```
// C#
public OracleNullValueException (string errMessage);
```

Parameters

- *`errMessage`*

  The specified error message.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleNullValueException Class
- OracleNullValueException Members

OracleNullValueException Static Methods

The `OracleNullValueException` static methods are listed in Table 5–137.

Table 5–137 OracleNullValueException Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleNullValueException Class
- OracleNullValueException Members

OracleNullValueException Properties

The `OracleNullValueException` properties are listed in Table 5–138.

Table 5–138 OracleNullValueException Properties

Properties	Description
`HelpLink`	Inherited from `Exception`
`InnerException`	Inherited from `Exception`
`Message`	Inherited from `OracleTypeException`
`Source`	Inherited from `OracleTypeException`
`StackTrace`	Inherited from `Exception`
`TargetSite`	Inherited from `Exception`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleNullValueException Class
- OracleNullValueException Members

OracleNullValueException Methods

The `OracleNullValueException` methods are listed in Table 5–139.

Table 5–139 OracleNullValueException Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetBaseException`	Inherited from `Exception`
`GetHashCode`	Inherited from `Object`
`GetObjectData`	Inherited from `Exception`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `OracleTypeException`

OracleTruncateException Class

The `OracleTruncateException` class represents an exception that is thrown when truncation in a ODP.NET Types class occurs.

Class Inheritance

```
Object
  Exception
    SystemException
      OracleTypeException
        OracleTruncateException
```

Declaration

```
// C#
public sealed class OracleTruncateException : OracleTypeException
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTruncateException Members
- OracleTruncateException Constructors
- OracleTruncateException Static Methods
- OracleTruncateException Properties
- OracleTruncateException Methods

OracleTruncateException Members

`OracleTruncateException` members are listed in the following tables:

OracleTruncateException Constructors

The `OracleTruncateException` constructors are listed in Table 5–140.

Table 5–140 OracleTruncateException Constructors

Constructor	Description
OracleTruncateException Constructors	Creates a new instance of the `OracleTruncateException` class (Overloaded)

OracleTruncateException Static Methods

The `OracleTruncateException` static methods are listed in Table 5–141.

Table 5–141 OracleTruncateException Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleTruncateException Properties

The `OracleTruncateException` properties are listed in Table 5–142.

Table 5–142 OracleTruncateException Properties

Properties	Description
`HelpLink`	Inherited from `Exception`
`InnerException`	Inherited from `Exception`
`Message`	Inherited from `OracleTypeException`
`Source`	Inherited from `OracleTypeException`
`StackTrace`	Inherited from `Exception`
`TargetSite`	Inherited from `Exception`

OracleTruncateException Methods

The `OracleTruncateException` methods are listed in Table 5–143.

Table 5–143 OracleTruncateException Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetBaseException`	Inherited from `Exception`
`GetHashCode`	Inherited from `Object`
`GetObjectData`	Inherited from `Exception`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `OracleTypeException`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTruncateException Class

OracleTruncateException Constructors

The `OracleTruncateException` constructors create new instances of the `OracleTruncateException` class

Overload List:

- OracleTruncateException()

  This constructor creates a new instance of the `OracleTruncateException` class with its default properties.

- OracleTruncateException(string)

  This constructor creates a new instance of the `OracleTruncateException` class with the specified error message, `errMessage`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTruncateException Class
- OracleTruncateException Members

OracleTruncateException()

This constructor creates a new instance of the `OracleTruncateException` class with its default properties.

Declaration

```
// C#
public OracleTruncateException();
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTruncateException Class
- OracleTruncateException Members

OracleTruncateException(string)

This constructor creates a new instance of the `OracleTruncateException` class with the specified error message, `errMessage`.

Declaration

```
// C#
public OracleTruncateException (string errMessage);
```

Parameters

- *`errMessage`*

  The specified error message.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTruncateException Class
- OracleTruncateException Members

OracleTruncateException Static Methods

The `OracleTruncateException` static methods are listed in Table 5–144.

Table 5–144 OracleTruncateException Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTruncateException Class
- OracleTruncateException Members

OracleTruncateException Properties

The `OracleTruncateException` properties are listed in Table 5–145.

Table 5–145 OracleTruncateException Properties

Properties	Description
`HelpLink`	Inherited from `Exception`
`InnerException`	Inherited from `Exception`
`Message`	Inherited from `OracleTypeException`
`Source`	Inherited from `OracleTypeException`
`StackTrace`	Inherited from `Exception`
`TargetSite`	Inherited from `Exception`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTruncateException Class
- OracleTruncateException Members

OracleTruncateException Methods

The `OracleTruncateException` methods are listed in Table 5–146.

Table 5–146 OracleTruncateException Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)
`GetBaseException`	Inherited from `Exception`
`GetHashCode`	Inherited from `Object`
`GetObjectData`	Inherited from `Exception`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `OracleTypeException`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleTruncateException Class
- OracleTruncateException Members

ODP.NET Type Objects

This section covers the following object classes:

- ODP.NET Types (ODP.NET LOB objects) consisting of these object classes:
  - OracleBFile Class
  - OracleBlob Class
  - OracleClob Class
- OracleRefCursor Class
- OracleXmlStream Class
- OracleXmlType Class

All offsets are `0`-based for all ODP.NET LOB and `OracleXmlStream` object parameters.

OracleBFile Class

An `OracleBFile` is an object that has a reference to `BFILE` data. It provides methods for performing operations on `BFiles`.

> **Note:** `OracleBFile` is supported for applications running against Oracle8.*x* and higher.

Class Inheritance

```
Object
  MarshalByRefObject
    Stream
      OracleBFile
```

Declaration

```
// C#
public sealed class OracleBFile : Stream, ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

`OracleBFile` is supported for applications running against Oracle8.*x* and higher.

Example

```
[C#]
  ...
  // assume:
  // 1. A valid connection is made
  // 2. contains a file c:\MyDir\MyFile.txt
  OracleBFile oraBFile = new OracleBFile(con, "c:\\MyDir", "MyFile.txt");

  // Open the oraBFile
  oraBFile.Open();

  // Read some data
```

```
...
int byteRead = oraBFile.Read(buffer, bufferOffset, amountToBeRead);

// Search for the 2nd occurrence of a byte pattern '123'
// from oraBFile starting at offset 1
byte[] pattern = new byte[3] { 1,2,3 };
int positionFound = oraBFile.Search(pattern, 1, 2);

// Close the BFile
oraBFile.CloseFile();
...
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Members
- OracleBFile Constructors
- OracleBFile Static Fields
- OracleBFile Static Methods
- OracleBFile Instance Properties
- OracleBFile Instance Methods

OracleBFile Members

`OracleBFile` members are listed in the following tables:

OracleBFile Constructors

`OracleBFile` constructors are listed in Table 5–147.

Table 5–147 OracleBFile Constructors

Constructor	Description
OracleBFile Constructors	Creates an instance of the `OracleBFile` class (Overloaded)

OracleBFile Static Fields

`OracleBFile` static fields are listed in Table 5–148.

Table 5–148 OracleBFile Static Fields

Field	Description
MaxSize	The static field holds the maximum number of bytes a `BFILE` can hold, which is 4,294,967,295 (2^32 - 1) bytes

OracleBFile Static Methods

`OracleBFile` static methods are listed in Table 5–149.

Table 5–149 OracleBFile Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleBFile Instance Properties

`OracleBFile` instance properties are listed in Table 5–150.

Table 5–150 OracleBFile Instance Properties

Properties	Description
CanRead	Indicates whether the LOB stream can be read
CanSeek	Indicates whether forward and backward seek operations can be performed
CanWrite	Indicates whether the LOB object supports writing
Connection	Indicates the connection used to read from a `BFILE`
DirectoryName	Indicates the directory alias of the `BFILE`
FileExists	Indicates whether or not the specified `BFILE` exists
FileName	Indicates the name of the `BFILE`
IsEmpty	Indicates whether the `BFILE` is empty or not
IsOpen	Indicates whether the `BFILE` has been opened by this instance or not

Table 5–150 OracleBFile Instance Properties (Cont.)

Properties	Description
Length	Indicates the size of the `BFILE` data in bytes
Position	Indicates the current read position in the LOB stream
Value	Returns the data, starting from the first byte in `BFILE`, as a byte array

OracleBFile Instance Methods

`OracleBFile` instance methods are listed in Table 5–151.

Table 5–151 OracleBFile Instance Methods

Methods	Description
`BeginRead`	Inherited from `Stream`
`BeginWrite`	*Not Supported*
Clone	Creates a copy of an `OracleBFile` object
Close	Closes the current stream and releases any resources associated with the stream
CloseFile	Closes the `BFILE` referenced by the current `BFILE` instance
Compare	Compares data referenced by the two `OracleBFiles`
`CreateObjRef`	Inherited from `MarshalByRefObject`
CopyTo	Copies data as specified (Overloaded)
Dispose	Releases resources allocated by this object
`EndRead`	Inherited from `Stream`
`EndWrite`	*Not Supported*
`Equals`	Inherited from `Object` (Overloaded)
`Flush`	*Not Supported*
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`

Table 5–151 OracleBFile Instance Methods (Cont.)

Methods	Description
InitializeLifetimeService	Inherited from MarshalByRefObject
IsEqual	Compares the LOB references
OpenFile	Opens the BFILE specified by the FileName and DirectoryName
Read	Reads a specified amount of bytes from the OracleBFile instance and populates the buffer
ReadByte	Inherited from Stream
Search	Searches for a binary pattern in the current instance of an OracleBFile
Seek	Sets the position on the current LOB stream
SetLength	*Not Supported*
ToString	Inherited from Object
Write	*Not Supported*
WriteByte	*Not Supported*

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Members

OracleBFile Constructors

OracleBFile constructors create new instances of the OracleBFile class.

Overload List:

- OracleBFile(OracleConnection)

  This constructor creates an instance of the OracleBFile class with an OracleConnection object.

- OracleBFile(OracleConnection, string, string)

  This constructor creates an instance of the OracleBFile class with an OracleConnection object, the location of the BFILE, and the name of the BFILE.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

OracleBFile(OracleConnection)

This constructor creates an instance of the `OracleBFile` class with an `OracleConnection` object.

Declaration

```
// C#
public OracleBFile(OracleConnection con);
```

Parameters

- *con*

  The `OracleConnection` object.

Exceptions

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The connection must be opened explicitly by the application. `OracleBFile` does not open the connection implicitly.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

OracleBFile(OracleConnection, string, string)

This constructor creates an instance of the `OracleBFile` class with an `OracleConnection` object, the location of the `BFILE`, and the name of the `BFILE`.

Declaration

```
// C#
```

```
public OracleBFile(OracleConnection con, string directoryName, string fileName);
```

Parameters

- *con*

  The OracleConnection object.

- *directoryName*

  The directory alias created by the CREATE DIRECTORY SQL statement.

- *fileName*

  The name of the external LOB.

Exceptions

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The OracleConnection must be opened explicitly by the application. OracleBFile does not open the connection implicitly.

To initialize a BFILE column using an OracleBFile instance as an input parameter of a SQL INSERT statement, *directoryName* and *fileName* must be properly set.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

OracleBFile Static Fields

OracleBFile static fields are listed in Table 5–152.

Table 5–152 OracleBFile Static Fields

Field	Description
MaxSize	The static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 (2^32 - 1) bytes

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

MaxSize

This static field holds the maximum number of bytes a `BFILE` can hold, which is 4,294,967,295 (2^32 - 1) bytes.

Declaration

```
// C#
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field is useful in code that checks whether the operation exceeds the maximum length allowed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

OracleBFile Static Methods

`OracleBFile` static methods are listed in Table 5–153.

Table 5–153 OracleBFile Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

OracleBFile Instance Properties

`OracleBFile` instance properties are listed in Table 5–154.

Table 5–154 OracleBFile Instance Properties

Properties	Description
CanRead	Indicates whether the LOB stream can be read
CanSeek	Indicates whether forward and backward seek operations can be performed
CanWrite	Indicates whether the LOB object supports writing
Connection	Indicates the connection used to read from a `BFILE`
DirectoryName	Indicates the directory alias of the `BFILE`
FileExists	Indicates whether or not the specified `BFILE` exists
FileName	Indicates the name of the `BFILE`
IsEmpty	Indicates whether the `BFILE` is empty or not
IsOpen	Indicates whether the `BFILE` has been opened by this instance or not
Length	Indicates the size of the `BFILE` data in bytes
Position	Indicates the current read position in the LOB stream
Value	Returns the data, starting from the first byte in `BFILE`, as a byte array

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CanRead

Overrides `Stream`

This instance property indicates whether the LOB stream can be read.

Declaration

```
// C#
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CanSeek

Overrides `Stream`

This instance property indicates whether forward and backward seek operations can be performed.

Declaration

```
// C#
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CanWrite

Overrides `Stream`

This instance property indicates whether the LOB object supports writing.

Declaration

```
// C#
public override bool CanWrite{get;}
```

Property Value

`BFILE` is read only.

Remarks

`BFILE` is read-only, therefore, the boolean value is always `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Connection

This instance property indicates the connection used to read from a `BFILE`.

Declaration

```
// C#
public OracleConnection Connection {get;}
```

Property Value

An object of `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

DirectoryName

This instance property indicates the directory alias of the BFILE.

Declaration

```
// C#
public string DirectoryName {get;set;}
```

Property Value

A string.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The value of the DirectoryName changed while the BFILE is open.

Remarks

The maximum length of a DirectoryName is 30 bytes.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

FileExists

This instance property indicates whether or not the BFILE specified by the DirectoryName and FileName exists.

Declaration

```
// C#
public bool FileExists {get;}
```

Property Value

bool

Exceptions

ObjectDisposedException - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

Unless a connection, file name, and directory name are provided, this property is set to `false` by default.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

FileName

This instance property indicates the name of the `BFILE`.

Declaration

```
// C#
public string FileName {get;set}
```

Property Value

A `string` that contains the `BFILE` name.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The value of the `DirectoryName` changed while the `BFILE` is open.

Remarks

The maximum length of a `FileName` is 255 bytes.

Changing the `FileName` property while the `BFILE` object is opened causes an exception.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

IsEmpty

This instance property indicates whether the `BFILE` is empty or not.

Declaration

```
// C#
public bool IsEmpty {get;}
```

Property Value

`bool`

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

IsOpen

This instance property indicates whether the `BFILE` has been opened by this instance or not.

Declaration

```
// C#
public bool IsOpen {get;}
```

Property Value

A `bool`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Length

Overrides `Stream`

This instance property indicates the size of the `BFILE` data in bytes.

Declaration

```
// C#
public override Int64 Length {get;}
```

Property Value

`Int64`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Position

Overrides `Stream`

This instance property indicates the current read position in the LOB stream.

Declaration

```
// C#
public override Int64 Position{get; set;}
```

Property Value

An `Int64` value that indicates the read position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The value is less than 0.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Value

This instance property returns the data, starting from the first byte in `BFILE`, as a byte array.

Declaration

```
// C#
public byte[] Value{get;}
```

Property Value

A byte array.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The length of data is bound by the maximum length of the byte array. The current value of the `Position` property is not used or changed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

OracleBFile Instance Methods

`OracleBFile` instance methods are listed in Table 5–155.

Table 5–155 OracleBFile Instance Methods

Methods	Description
`BeginRead`	Inherited from `Stream`
`BeginWrite`	*Not Supported*
Clone	Creates a copy of an `OracleBFile` object
Close	Closes the current stream and releases any resources associated with the stream
CloseFile	Closes the `BFILE` referenced by the current `BFILE` instance
Compare	Compares data referenced by the two `OracleBFiles`
`CreateObjRef`	Inherited from `MarshalByRefObject`
CopyTo	Copies data as specified (Overloaded)
Dispose	Releases resources allocated by this object
`EndRead`	Inherited from `Stream`
`EndWrite`	*Not Supported*
`Equals`	Inherited from `Object` (Overloaded)
`Flush`	*Not Supported*
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
IsEqual	Compares the LOB references

Table 5–155 OracleBFile Instance Methods (Cont.)

Methods	Description
OpenFile	Opens the `BFILE` specified by the `FileName` and `DirectoryName`
Read	Reads a specified amount of bytes from the `OracleBFile` instance and populates the `buffer`
`ReadByte`	Inherited from `Stream`
Search	Searches for a binary pattern in the current instance of an `OracleBFile`
Seek	Sets the position on the current LOB stream
`SetLength`	*Not Supported*
`ToString`	Inherited from `Object`
`Write`	*Not Supported*
`WriteByte`	*Not Supported*

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Clone

This instance method creates a copy of an `OracleBFile` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleBFile` object.

Implements

`ICloneable`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
//Need a proper casting for the return value when cloned
OracleBFile oraBfile_cloned = (OracleBFile) oraBfile.Clone();
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Close

Overrides `Stream`

This instance method closes the current stream and releases any resources associated with it.

Declaration

```
// C#
public override void Close();
```

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CloseFile

This instance method closes the `BFILE` referenced by the current `BFILE` instance.

Declaration

```
// C#
public void CloseFile();
```

Remarks

No error is returned if the `BFILE` exists, but is not opened.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Compare

This instance method compares data referenced by the two `OracleBFiles`.

Declaration

```
// C#
public int Compare(Int64 src_offset, OracleBFile obj, Int64 dst_offset, Int64
amount);
```

Parameters

- *src_offset*

  The offset of the current instance.

- *obj*

  The provided `OracleBFile` object.

- *dst_offset*

The offset of the `OracleBFile` object.

- *`amount`*

  The number of bytes to compare.

Return Value

Returns a number that is:

- Less than zero: if the `BFILE` data of the current instance is less than that of the provided `BFILE` data.
- Zero: if both the `BFILE`s store the same data.
- Greater than zero: if the `BFILE` data of the current instance is greater than that of the provided `BFILE` data.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* is less than `0`.

Remarks

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

The `BFILE` needs to be opened using `OpenFile` before the operation.

Example

```
// C#
...
// Assume you have 2 valid files in C:\MyDir
OracleBFile myBFile1 = new OracleBFile(con, "c:\\MyDir", "MyFile1.txt");
OracleBFile myBFile2 = new OracleBFile(con, "c:\\MyDir", "MyFile2.txt");

int src_offset = 10;
int dst_offset = 20;
int amount     = 5;

int result = myBFile1.Compare(src_offset, myBFile2, dst_offset, amount);
```

```
if ( result == 0 )
  Console.WriteLine("Identical");
else
  Console.WriteLine("Not Identical");
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CopyTo

`CopyTo` copies data from the current instance to the provided object.

Overload List:

- CopyTo(OracleBlob)

  This instance method copies data from the current instance to the provided `OracleBlob` object.

- CopyTo(OracleBlob, Int64)

  This instance method copies data from the current `OracleBFile` instance to the provided `OracleBlob` object with the specified destination offset.

- CopyTo(Int64, OracleBlob, Int64, Int64)

  This instance method copies data from the current `OracleBFile` instance to the provided `OracleBlob` object with the specified source offset, destination offset, and character amounts.

- CopyTo(OracleClob)

  This instance method copies data from the current `OracleBFile` instance to the provided `OracleClob` object.

- CopyTo(OracleClob, Int64)

  This instance method copies data from the current `OracleBFile` instance to the provided `OracleClob` object with the specified destination offset.

- CopyTo(Int64, OracleClob, Int64, Int64)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleClob` object with the specified source offset, destination offset, and amount of characters.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CopyTo(OracleBlob)

This instance method copies data from the current instance to the provided `OracleBlob` object.

Declaration

```
// C#
public Int64 CopyTo(OracleBlob obj);
```

Parameters

- *obj*

  The `OracleBlob` object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CopyTo(OracleBlob, Int64)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleBlob` object with the specified destination offset.

Declaration

```
// C#
public Int64 CopyTo(OracleBlob obj, Int64 dst_offset);
```

Parameters

- *obj*

  The `OracleBlob` object to which the data is copied.

- *dst_offset*

  The offset (in bytes) at which the `OracleBlob` object is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *dst_offset* is less than `0`.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same OracleConnection object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CopyTo(Int64, OracleBlob, Int64, Int64)

This instance method copies data from the current OracleBFile instance to the provided OracleBlob object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset,OracleBlob obj,Int64 dst_offset, Int64
amount);
```

Parameters

- *src_offset*

  The offset (in bytes) in the current instance, from which the data is read.

- *obj*

  An OracleBlob object to which the data is copied.

- *dst_offset*

  The offset (in bytes) to which the OracleBlob object is copied.

- *amount*

  The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

ObjectDisposedException - The object is already disposed.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* is less than `0`.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CopyTo(OracleClob)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleClob` object.

Declaration

```
// C#
public Int64 CopyTo(OracleClob obj);
```

Parameters

- *`obj`*

  The `OracleClob` object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CopyTo(OracleClob, Int64)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleClob` object with the specified destination offset.

Declaration

```
// C#
public Int64 CopyTo(OracleClob obj, Int64 dst_offset);
```

Parameters

- *obj*

  The `OracleClob` object that the data is copied to.

- *dst_offset*

  The offset (in characters) at which the `OracleClob` object is copied to.

Return Value

The amount copied.

Exceptions

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *`dst_offset`* is less than `0`.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleClob` data, spaces are written into the `OracleClob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

CopyTo(Int64, OracleClob, Int64, Int64)

This instance method copies data from the current `OracleBFile` instance to the provided `OracleClob` object with the specified source offset, destination offset, and amount of characters.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset,OracleClob obj,Int64 dst_offset,Int64
amount);
```

Parameters

- *`src_offset`*

The offset (in characters) in the current instance, from which the data is read.

- *obj*

  An OracleClob object that the data is copied to.

- *dst_offset*

  The offset (in characters) at which the OracleClob object is copied to.

- *amount*

  The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentOutOfRangeException - The *src_offset*, the *dst_offset*, or the *amount* is less than 0.

InvalidOperationException - This exception is thrown if any of the following conditions exist:

- The OracleConnection is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the current OracleClob data, spaces are written into the OracleClob until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Dispose

This instance method releases resources allocated by this object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls may lead to exceptions. The object cannot be reused after being disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

IsEqual

This instance method compares the LOB references.

Declaration

```
// C#
public bool IsEqual(OracleBFile obj);
```

Parameters

- *obj*

  The provided `OracleBFile` object.

Return Value

Returns `true` if the current `OracleBFile` and the provided `OracleBFile` object refer to the same external LOB. Returns `false` otherwise.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

Note that this method can return `true` even if the two `OracleBFile` objects return `false` for `==` or `Equals()` since two different `OracleBFile` instances can refer to the same external LOB.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

OpenFile

This instance method opens the `BFILE` specified by the `FileName` and `DirectoryName`.

Declaration

```
// C#
public void OpenFile();
```

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

Many operations, such as `Compare()`, `CopyTo()`, `Read()`, and `Search()` require that the `BFILE` be opened using `OpenFile` before the operation.

Calling `OpenFile` on an opened `BFILE` is not operational.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Read

Overrides `Stream`

This instance method reads a specified amount of bytes from the `OracleBFile` instance and populates the `buffer`.

Declaration

```
// C#
public override int Read(byte[ ] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  The byte array buffer to be populated.

- *`offset`*

  The offset of the byte array buffer to be populated.

- *`count`*

  The amount of bytes to read.

Return Value

The return value indicates the number of bytes read from the `BFILE`, that is, the external LOB.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - Either the *`offset`* or the *`count`* parameter is less than `0` or the *`offset`* is greater than or equal to the *`buffer`*.`Length` or the *`offset`* and the *`count`* together are greater than *`buffer`*.`Length`.

Remarks

The LOB data is read starting from the position specified by the `Position` property.

Example

```
// C#
...
byte buffer         = new byte[1024];
int  bufferOffset   = 10;
int  amountToBeRead = 10;

// Read some data
int byteRead = oraBFile.Read(buffer, bufferOffset, amountToBeRead);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Search

This instance method searches for a binary pattern in the current instance of an `OracleBFile`.

Declaration

```
// C#
public int Search(byte[ ] val, Int64 offset, Int64 nth);
```

Parameters

- *val*

  The binary pattern being searched for.

- *offset*

  The `0`-based offset (in bytes) starting from which the `OracleBFile` is searched.

- *nth*

  The specific occurrence (`1`-based) of the match for which the offset is returned.

Return Value

Returns the absolute *offset* of the start of the matched pattern (in bytes) for the *nth* occurrence of the match. Otherwise, 0 is returned.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - Either the *offset* is less than 0 or *nth* is less than or equal to 0 or *val*.Length is greater than 16383 or *nth* is greater than or equal to OracleBFile.MaxSize or *offset* is greater than or equal to OracleBFile.MaxSize.

Remarks

The limit of the search pattern is 16383 bytes.

Example

```
// C#
...
// Search for the 2nd occurrence of a byte pattern '123'
// from oraBFile starting at offset 1
byte[] pattern = new byte[3] { 1,2,3 };
int positionFound = oraBFile.Search(pattern, 1, 2);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

Seek

Overrides Stream

This instance method sets the position on the current LOB stream.

Declaration

```
// C#
```

```
public override int64 Seek(Int64 offset, SeekOrigin origin);
```

Parameters

- *offset*

  A byte offset relative to origin.

- *origin*

  A value of type `System.IO.SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

Returns an `Int64` that indicates the position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

If *offset* is negative, the new position precedes the position specified by *origin* by the number of bytes specified by *offset*.

If *offset* is zero, the new position is the position specified by *origin*.

If *offset* is positive, the new position follows the position specified by *origin* by the number of bytes specified by *offset*.

`SeekOrigin.Begin` specifies the beginning of a stream.

`SeekOrigin.Current` specifies the current position within a stream.

`SeekOrigin.End` specifies the end of a stream.

Example

```
// C#
 ...
// Set the Position to 5 bytes (with respect to SeekOrigin.Begin), read 10 bytes
// out, and put the data in a buffer with offset = 10 bytes
byte buffer         = new byte[1024];
int  bufferOffset   = 10;
int  amountToBeRead = 10;
// Seek
```

```
int newPosition = oraBFile.Seek(5, SeekOrigin.Begin);

// Read some data
int byteRead = oraBFile.Read(buffer, bufferOffset, amountToBeRead);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBFile Class
- OracleBFile Members

OracleBlob Class

An OracleBlob object is an object that has a reference to BLOB data. It provides methods for performing operations on BLOBs.

Class Inheritance

```
Object
  MarshalByRefObject
    Stream
      OracleBlob
```

Declaration

```
// C#
public sealed class OracleBlob : Stream, ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
[C#]
...
// assume: A valid connection is made
OracleBlob oraBlob = new OracleBlob(con);

// Read some data
...
int byteRead = oraBlob.Read(buffer, bufferOffset, amountToBeRead);

// Search for the 2nd occurrence of a byte pattern '123'
// from the oraBlob starting at offset 1
byte[] pattern = new byte[3] { 1,2,3 };
int positionFound = oraBlob.Search(pattern, 1, 2);

// Append 2 bytes {4,5} to the oraBlob
oraBlob.Append(new byte[3] {4,5,6}, 1, 2);

// Write 64 bytes, starting at buffer offset 512
byte[4096] buffer = new byte[4096];
```

```
...
oraBlob.Write(buffer, 512, 64);

// Erase 64 bytes of data starting at offset=1024
oraBlob.Erase(1024,64);
...
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Members
- OracleBlob Constructors
- OracleBlob Static Fields
- OracleBlob Static Methods
- OracleBlob Instance Properties
- OracleBlob Instance Methods

OracleBlob Members

`OracleBlob` members are listed in the following tables:

OracleBlob Constructors

`OracleBlob` constructors are listed in Table 5–156.

Table 5–156 OracleBlob Constructors

Constructor	Description
OracleBlob Constructors	Creates an instance of the `OracleBlob` class (Overloaded)

OracleBlob Static Fields

`OracleBlob` static fields are listed in Table 5–157.

Table 5–157 OracleBlob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a BLOB can hold, which is 4,294,967,295 (2^32 - 1) bytes

OracleBlob Static Methods

OracleBlob static methods are listed in Table 5–158.

Table 5–158 OracleBlob Static Methods

Methods	Description
Equals	Inherited from Object (Overloaded)

OracleBlob Instance Properties

OracleBlob instance properties are listed in Table 5–159.

Table 5–159 OracleBlob Instance Properties

Properties	Description
CanRead	Indicates whether the LOB stream can be read
CanSeek	Indicates whether forward and backward seek operations be performed
CanWrite	Indicates whether the LOB object supports writing
Connection	Indicates the OracleConnection that is used to retrieve and write BLOB data
IsEmpty	Indicates whether the BLOB is empty or not
IsInChunkWriteMode	Indicates whether the BLOB has been opened to defer index updates
IsTemporary	Indicates whether or not the current instance is bound to a temporary BLOB
Length	Indicates the size of the BLOB data
OptimumChunkSize	Indicates the minimum number of bytes to retrieve or send from the server during a read or write operation
Position	Indicates the current read or write position in the LOB stream

Table 5–159 OracleBlob Instance Properties (Cont.)

Properties	Description
Value	Returns the data, starting from the first byte in `BLOB`, as a byte array

OracleBlob Instance Methods

`OracleBlob` instance methods are listed in Table 5–160.

Table 5–160 OracleBlob Instance Methods

Methods	Description
Append	Appends the supplied data to the current `OracleBlob` instance (Overloaded)
BeginChunkWrite	Opens the `BLOB`
`BeginRead`	Inherited from `Stream`
`BeginWrite`	Inherited from `Stream`
Clone	Creates a copy of an `OracleBlob` object
Close	Closes the current stream and releases any resources associated with it
Compare	Compares data referenced by the current instance and that of the supplied object
CopyTo	Copies from the current `OracleBlob` instance to an `OracleBlob` object (Overloaded)
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Releases resources allocated by this `object`
EndChunkWrite	Closes the `BLOB` referenced by the current `OracleBlob` instance
`EndRead`	Inherited from `Stream`
`EndWrite`	Inherited from `Stream`
`Equals`	Inherited from `Object` (Overloaded)
Erase	Erases data (Overloaded)
`Flush`	*Not supported*
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`

Table 5–160 OracleBlob Instance Methods (Cont.)

Methods	Description
GetType	Inherited from Object
InitializedLifetimeService	Inherited from MarshalByRefObject
IsEqual	Compares the LOB data referenced by the two OracleBlobs
Read	Reads a specified amount of bytes from the ODP.NET LOB Type instance and populates the buffer
ReadByte	Inherited from Stream
Search	Searches for a binary pattern in the current instance of an OracleBlob
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the BLOB value to the specified length
ToString	Inherited from Object
Write	Writes the supplied buffer into the OracleBlob
WriteByte	Inherited from Stream

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Members

OracleBlob Constructors

OracleBlob constructors are listed in Table 5–156.

Overload List:

- OracleBlob(OracleConnection)

  This constructor creates an instance of the OracleBlob class bound to a temporary BLOB with an OracleConnection object.

- OracleBlob(OracleConnection, bool)

This constructor creates an instance of the OracleBlob class bound to a temporary BLOB with an OracleConnection object and a boolean value for caching.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

OracleBlob(OracleConnection)

This constructor creates an instance of the OracleBlob class bound to a temporary BLOB with an OracleConnection object.

Declaration

```
// C#
public OracleBlob(OracleConnection con);
```

Parameters

- *con*

  The OracleConnection object.

Exceptions

InvalidOperationException - The OracleConnection is not opened.

Remarks

The connection must be opened explicitly by the application. OracleBlob does not open the connection implicitly.

The temporary BLOB utilizes the provided connection to store BLOB data. Caching is not turned on by this constructor.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

OracleBlob(OracleConnection, bool)

This constructor creates an instance of the `OracleBlob` class bound to a temporary `BLOB` with an `OracleConnection` object and a boolean value for caching.

Declaration

```
// C#
public OracleBlob(OracleConnection con, bool bCaching);
```

Parameters

- *con*

  The `OracleConnection` object.

- *bCaching*

  A flag for enabling or disabling server-side caching.

Exceptions

`InvalidOperationException` - The `OracleConnection` is not opened.

Remarks

The connection must be opened explicitly by the application. `OracleBlob` does not open the connection implicitly.

The temporary `BLOB` uses the provided connection to store `BLOB` data. The *bCaching* input parameter determines whether or not server-side caching is used.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

OracleBlob Static Fields

`OracleBlob` static fields are listed in Table 5–161.

Table 5–161 OracleBlob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a `BLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

MaxSize

The `MaxSize` field holds the maximum number of bytes a `BLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes.

Declaration

```
// C#
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field can be useful in code that checks whether the operation exceeds the maximum length allowed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

OracleBlob Static Methods

`OracleBlob` static methods are listed in Table 5–162.

Table 5–162 OracleBlob Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

OracleBlob Instance Properties

`OracleBlob` instance properties are listed in Table 5–163.

Table 5–163 OracleBlob Instance Properties

Properties	Description
CanRead	Indicates whether the LOB stream can be read
CanSeek	Indicates whether forward and backward seek operations be performed
CanWrite	Indicates whether the LOB object supports writing
Connection	Indicates the `OracleConnection` that is used to retrieve and write `BLOB` data
IsEmpty	Indicates whether the `BLOB` is empty or not
IsInChunkWriteMode	Indicates whether the `BLOB` has been opened to defer index updates
IsTemporary	Indicates whether or not the current instance is bound to a temporary `BLOB`
Length	Indicates the size of the `BLOB` data
OptimumChunkSize	Indicates the minimum number of bytes to retrieve or send from the server during a read or write operation
Position	Indicates the current read or write position in the LOB stream
Value	Returns the data, starting from the first byte in `BLOB`, as a byte array

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

CanRead

Overrides `Stream`

This instance property indicates whether the LOB stream can be read.

Declaration

```
// C#
```

```
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

CanSeek

Overrides `Stream`

This instance property indicates whether forward and backward seek operations can be performed.

Declaration

```
// C#
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

CanWrite

Overrides `Stream`

This instance property indicates whether the LOB object supports writing.

Declaration

```
// C#
public override bool CanWrite{get;}
```

Property Value

If the LOB stream can be written, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Connection

This instance property indicates the `OracleConnection` that is used to retrieve and write `BLOB` data.

Declaration

```
// C#
public OracleConnection Connection {get;}
```

Property Value

An object of `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

IsEmpty

This instance property indicates whether the `BLOB` is empty or not.

Declaration

```
// C#
public bool IsEmpty {get;}
```

Property Value

A `bool` that indicates whether the `BLOB` is empty.

Exceptions

`ObjectDisposedException` - The object is already disposed.

IsInChunkWriteMode

This instance property indicates whether the `BLOB` has been opened to defer index updates.

Declaration

```
// C#
public bool IsInChunkWriteMode{get;}
```

Property Value

If the `BLOB` has been opened, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

IsTemporary

This instance property indicates whether or not the current instance is bound to a temporary `BLOB`.

Declaration

```
// C#
public bool IsTemporary {get;}
```

Property Value

`bool`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Length

Overrides `Stream`

This instance property indicates the size of the `BLOB` data in bytes.

Declaration

```
// C#
public override Int64 Length {get;}
```

Property Value

A number indicating the size of the `BLOB` data in bytes.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

OptimumChunkSize

This instance property indicates the minimum number of bytes to retrieve or send from the server during a read or write operation.

Declaration

```
// C#
public int OptimumChunkSize{get;}
```

Property Value

A number representing the minimum bytes to retrieve or send.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Position

Overrides `Stream`

This instance property indicates the current read or write position in the LOB stream.

Declaration

```
// C#
public override Int64 Position{get; set;}
```

Property Value

An `Int64` that indicates the read or write position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The `Position` is less than `0`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Value

This instance property returns the data, starting from the first byte in the `BLOB`, as a byte array.

Declaration

```
// C#
public Byte[] Value{get;}
```

Property Value

A byte array.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The `Value` is less than `0`.

Remarks

The value of `Position` is not used or changed by using this property. 2 GB is the maximum byte array length that can be returned by this property.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

OracleBlob Instance Methods

`OracleBlob` instance methods are listed in Table 5–164.

Table 5–164 OracleBlob Instance Methods

Methods	Description
Append	Appends the supplied data to the current `OracleBlob` instance (Overloaded)
BeginChunkWrite	Opens the `BLOB`
`BeginRead`	Inherited from `Stream`
`BeginWrite`	Inherited from `Stream`
Clone	Creates a copy of an `OracleBlob` object
Close	Closes the current stream and releases any resources associated with it
Compare	Compares data referenced by the current instance and that of the supplied object

Table 5–164 OracleBlob Instance Methods (Cont.)

Methods	Description
CopyTo	Copies from the current `OracleBlob` instance to an `OracleBlob` object (Overloaded)
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Releases resources allocated by this `object`
EndChunkWrite	Closes the `BLOB` referenced by the current `OracleBlob` instance
`EndRead`	Inherited from `Stream`
`EndWrite`	Inherited from `Stream`
`Equals`	Inherited from `Object` (Overloaded)
Erase	Erases data (Overloaded)
`Flush`	*Not supported*
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializedLifetimeService`	Inherited from `MarshalByRefObject`
IsEqual	Compares the LOB data referenced by the two `OracleBlobs`
Read	Reads a specified amount of bytes from the ODP.NET LOB Type instance and populates the `buffer`
`ReadByte`	Inherited from `Stream`
Search	Searches for a binary pattern in the current instance of an `OracleBlob`
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the `BLOB` value to the specified length
`ToString`	Inherited from `Object`
Write	Writes the supplied buffer into the `OracleBlob`
`WriteByte`	Inherited from `Stream`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Append

`Append` appends the supplied data to the end of the current `OracleBlob` instance.

Overload List:

- Append(OracleBlob)

  This instance method appends the `BLOB` data referenced by the provided `OracleBlob` object to the current `OracleBlob` instance.

- Append(byte[ ], int, int)

  This instance method appends data from the supplied byte array buffer to the end of the current `OracleBlob` instance.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Append(OracleBlob)

This instance method appends the `BLOB` data referenced by the provided `OracleBlob` object to the current `OracleBlob` instance.

Declaration

```
// C#
public void Append(OracleBlob obj);
```

Parameters

- *obj*

  An object of `OracleBlob`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

Remarks

No character set conversions are made.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Append(byte[ ], int, int)

This instance method appends data from the supplied byte array buffer to the end of the current `OracleBlob` instance.

Declaration

```
// C#
public void Append(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*

  An array of bytes.

- *offset*

  The zero-based byte offset in the buffer from which data is read.

- *count*

  The number of bytes to be appended.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Example

```
// C#
...
// Append 2 bytes {4,5} to the oraBlob
oraBlob.Append(new byte[3] {4,5,6}, 1, 2);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

BeginChunkWrite

This instance method opens the `BLOB`.

Declaration

```
// C#
public void BeginChunkWrite();
```

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

`BeginChunkWrite` does not need to be called before manipulating the BLOB data. This is provided for performance reasons.

After this method is called, write operations do not cause the domain or function-based index on the column to be updated. Index updates occur only once after `EndChunkWrite` is called.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Clone

This instance method creates a copy of an `OracleBlob` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleBlob` object.

Implements

`ICloneable`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
//Need a proper casting for the return value when cloned
OracleBlob oraBlob_cloned = (OracleBlob) oraBlob.Clone();
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Close

Overrides `Stream`

This instance method closes the current stream and releases any resources associated with it.

Declaration

```
// C#
public override void Close();
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Compare

This instance method compares data referenced by the current instance and that of the supplied object.

Declaration

```
// C#
public int Compare(Int64 src_offset, OracleBlob obj, Int64 dst_offset,
 Int64 amount);
```

Parameters

- *src_offset*

  The comparison starting point (in bytes) for the current instance.

- *obj*

  The provided `OracleBlob` object.

- *dst_offset*

The comparison starting point (in bytes) for the provided `OracleBlob`.

- *`amount`*

  The number of bytes to compare.

Return Value

Returns a value that is:

- Less than zero: if the data referenced by the current instance is less than that of the supplied instance
- Zero: if both objects reference the same data
- Greater than zero: if the data referenced by the current instance is greater than that of the supplied instance

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* parameter is less than `0`.

Remarks

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

CopyTo

`CopyTo` copies data from the current instance to the provided `OracleBlob` object.

Overload List:

- CopyTo(OracleBlob)

This instance method copies data from the current instance to the provided `OracleBlob` object.

- CopyTo(OracleBlob, Int64)

  This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified destination offset.

- CopyTo(Int64, OracleBlob, Int64, Int64)

  This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified source offset, destination offset, and character amounts.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

CopyTo(OracleBlob)

This instance method copies data from the current instance to the provided `OracleBlob` object.

Declaration

```
// C#
public Int64 CopyTo(OracleBlob obj);
```

Parameters

- *obj*

  The `OracleBlob` object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

CopyTo(OracleBlob, Int64)

This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified destination offset.

Declaration

```
// C#
public Int64 CopyTo(OracleBlob obj, Int64 dst_offset);
```

Parameters

- *obj*

  The `OracleBlob` object to which the data is copied.

- *dst_offset*

  The offset (in bytes) at which the `OracleBlob` object is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *dst_offset* is less than `0`.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The OracleConnection is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *dst_offset* is beyond the end of the OracleBlob data, spaces are written into the OracleBlob until the *dst_offset* is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same OracleConnection object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

CopyTo(Int64, OracleBlob, Int64, Int64)

This instance method copies data from the current OracleBlob instance to the provided OracleBlob object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset,OracleBlob obj,Int64 dst_offset,Int64
amount);
```

Parameters

- *src_offset*

  The offset (in bytes) in the current instance, from which the data is read.

- *obj*

  The OracleBlob object to which the data is copied.

- *dst_offset*

  The offset (in bytes) at which the OracleBlob object is copied.

- *amount*

The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* parameter is less than `0`.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleBlob` data, spaces are written into the `OracleBlob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

Example

```
// C#
...
// Assume you have 2 valid blobs
OracleBlob src_blob = new OracleBlob(con);
OracleBlob target_blob = new OracleBlob(con);

// Copy 1024 bytes from src_blob (begin at offset 10) to target_blob
// (starting at offset 5)
src_blob.CopyTo(10, target_blob, 5, 1024);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Dispose

This instance method releases resources allocated by this object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

Once `Dispose()` is called, the object of `OracleBlob` is in an uninitialized state.

Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls may lead to exceptions. The object cannot be reused after being disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

EndChunkWrite

This instance method closes the `BLOB` referenced by the current `OracleBlob` instance.

Declaration

```
// C#
public void EndChunkWrite();
```

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

Index updates occur immediately if there is write operation(s) deferred by the `BeginChunkWrite` method.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Erase

`Erase` erases a portion or all data.

Overload List:

- Erase()

  This instance method erases all data.

- Erase(Int64, Int64)

  This instance method erases a specified portion of data.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Erase()

This instance method erases all data.

Declaration

```
// C#
public Int64 Erase();
```

Return Value

The number of bytes erased.

Remarks

`Erase()` replaces all data with zero-byte fillers.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Erase(Int64, Int64)

This instance method erases a specified portion of data.

Declaration

```
// C#
public Int64 Erase(Int64 offset, Int64 amount);
```

Parameters

- *offset*

  The offset from which to erase.

- *amount*

  The quantity (in bytes) to erase.

Return Value

The number of bytes erased.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The *offset* or *amount* parameter is less than `0`.

Remarks

Replaces the specified *amount* of data with zero-byte fillers.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

IsEqual

This instance method compares the LOB data referenced by the two `OracleBlobs`.

Declaration

```
// C#
public bool IsEqual(OracleBlob obj);
```

Parameters

- *obj*

  An `OracleBlob` object.

Return Value

If the current `OracleBlob` and the provided `OracleBlob` refer to the same LOB, returns `true`. Returns `false` otherwise.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

Note that this method can return `true` even if the two `OracleBlob` objects return `false` for == or `Equals()` because two different `OracleBlob` instances can refer to the same LOB.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Read

Overrides `Stream`

This instance method reads a specified amount of bytes from the ODP.NET LOB instance and populates the `buffer`.

Declaration

```
// C#
public override int Read(byte[ ] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  The byte array buffer to be populated.

- *`offset`*

  The starting offset (in bytes) at which the buffer is populated.

- *`count`*

  The amount of bytes to read.

Return Value

The return value indicates the number of bytes read from the LOB.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *`offset`* or the *`count`* parameter is less than `0`.
- The *`offset`* is greater than or equal to the *`buffer`*.`Length`.

- The *offset* and the *count* together are greater than the *buffer*.Length.

Remarks

The LOB data is read starting from the position specified by the Position property.

Example

```
// C#
...
byte buffer         = new byte[1024];
int  bufferOffset   = 10;
int  amountToBeRead = 10;
// Read some data
...
int byteRead = oraBlob.Read(buffer, bufferOffset, amountToBeRead);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Search

This instance method searches for a binary pattern in the current instance of an OracleBlob.

Declaration

```
// C#
public Int64 Search(byte[] val, int64 offset, int64 nth);
```

Parameters

- *val*

  The binary pattern being searched for.

- *offset*

  The 0-based offset (in bytes) starting from which the OracleBlob is searched.

- *nth*

The specific occurrence (1-based) of the match for which the absolute offset (in bytes) is returned.

Return Value

Returns the absolute *offset* of the start of the matched pattern (in bytes) for the *nth* occurrence of the match. Otherwise, 0 is returned.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The *offset* is less than 0.
- The *nth* is less than or equal to 0.
- The *val*.Length is greater than 16383.
- The *nth* is greater than or equal to OracleBlob.MaxSize.
- The *offset* is greater than or equal to OracleBlob.MaxSize.

Remarks

The limit of the search pattern is 16383 bytes.

Example

```
// C#
...
// Search for the 2nd occurrence of a byte pattern '123'
// from the oraBlob starting at offset 1
byte[] pattern = new byte[3] { 1,2,3 };
int positionFound = oraBlob.Search(pattern, 1, 2);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Seek

Overrides `Stream`

This instance method sets the position on the current LOB stream.

Declaration

```
// C#
public override Int64(Int64 offset, SeekOrigin origin);
```

Parameters

- *offset*

  A byte offset relative to origin.

- *origin*

  A value of type `System.IO.SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

Returns `Int64` for the position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

If *offset* is negative, the new position precedes the position specified by *origin* by the number of bytes specified by `offset`.

If *offset* is zero, the new position is the position specified by *origin*.

If *offset* is positive, the new position follows the position specified by *origin* by the number of bytes specified by *offset*.

`SeekOrigin.Begin` specifies the beginning of a stream.

`SeekOrigin.Current` specifies the current position within a stream.

`SeekOrigin.End` specifies the end of a stream.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

SetLength

Overrides `Stream`

This instance method trims or truncates the `BLOB` value to the specified length (in bytes).

Declaration

```
// C#
public override void SetLength(Int64 newlen);
```

Parameters

- *`newlen`*

  The desired length of the current stream in bytes.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The *`newlen`* parameter is less than `0`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

Write

Overrides `Stream`

This instance method writes the supplied buffer into the `OracleBlob`.

Declaration

```
// C#
public override void Write(byte[ ] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  The byte array `buffer` that provides the data.

- *`offset`*

  The `0`-based offset (in bytes) from which the `buffer` is read.

- *`count`*

  The amount of data (in bytes) that is to be written into the `OracleBlob`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *`offset`* or the *`count`* is less than `0`.
- The *`offset`* is greater than or equal to the *`buffer`*.`Length`.
- The *`offset`* and the *`count`* together are greater than *`buffer`*.`Length`.

Remarks

Destination *`offset`* in the `OracleBlob` can be specified by the `Position` property.

Example

```
// C#
...
// Begin ChunkWrite to improve performance
```

```
// Index updates occur only once after EndChunkWrite
oraBlob.BeginChunkWrite();

// Set the write from the beginning;
oraBlob.Position = 0;

// Write to the oraBlob in chunks of 10, each 1024 bytes
for ( int i=0; i<10; i++ )
{
  byte[1024] b;
  b = b[0];    // some new value to be written
  oraBlob.Write(b, 0, b.Length);
}

oraBlob.EndChunkWrite();
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleBlob Class
- OracleBlob Members

OracleClob Class

An OracleClob is an object that has a reference to CLOB data. It provides methods for performing operations on CLOBs.

> **Note:** The OracleClob object uses the client side character set when retrieving or writing CLOB data using a .NET Framework byte array.

Class Inheritance

```
Object
  MarshalByRefObject
    Stream
      OracleClob
```

Declaration

```
// C#
public sealed class OracleClob : Stream, ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
// Example demostrates Read, Append, Search, Write and, Erase APIs
// assume: A valid connection 'con'is created
OracleClob oraClob = new OracleClob(con);

// Read some data (in characters)
char buffer = new char[1024];
int  bufferOffset  = 10;
int  amountToBeRead = 10;

int charRead = oraClob.Read(buffer, bufferOffset, amountToBeRead);

// Search for the 2nd occurrence of a char pattern 'oracle'
// from the oraClob starting at offset 1
```

```
char[6] pattern = new char[6] { "o", "r", "a", "c", "l", "e" };
int positionFound = oraClob.Search(pattern, 1, 2);

// Append 2 char to the oraClob
oraClob.Append(new char[3] {"f", "o", "o"}, 1, 2);

// Write 32 char, starting at buffer offset 512
char[4096] buffer = new char[4096];
...
oraClob.Write(buffer, 512, 32);

// Erase 64 char of data starting at offset=1024
oraClob.Erase(1024,64);
...
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Members
- OracleClob Constructors
- OracleClob Static Fields
- OracleClob Static Methods
- OracleClob Instance Properties
- OracleClob Instance Methods

OracleClob Members

`OracleClob` members are listed in the following tables:

OracleClob Constructors

`OracleClob` constructors are listed in Table 5–165.

Table 5–165 OracleClob Constructors

Constructor	Description
OracleClob Constructors	Creates an instance of the `OracleClob` class bound to a temporary `CLOB` (Overloaded)

OracleClob Static Fields

`OracleClob` static fields are listed in Table 5–166.

Table 5–166 OracleClob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a `CLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes

OracleClob Static Methods

`OracleClob` static methods are listed in Table 5–167.

Table 5–167 OracleClob Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleClob Instance Properties

`OracleClob` instance properties are listed in Table 5–168.

Table 5–168 OracleClob Instance Properties

Properties	Description
CanRead	Indicates whether the LOB stream can be read
CanSeek	Indicates whether forward and backward seek operations can be performed
CanWrite	Indicates whether the LOB stream can be written
Connection	Indicates the `OracleConnection` that is used to retrieve and write `CLOB` data
IsEmpty	Indicates whether the `CLOB` is `empty` or not
IsInChunkWriteMode	Indicates whether or not the `CLOB` has been opened

Table 5–168 OracleClob Instance Properties (Cont.)

Properties	Description
IsNCLOB	Indicates whether the `OracleClob` object represents an `NCLOB`.
IsTemporary	Indicates whether or not the current instance is bound to a temporary `CLOB`
Length	Indicates the size of the `CLOB` data in bytes
OptimumChunkSize	Indicates the minimum number of bytes to retrieve or send from the server during a read or write operation
Position	Indicates the current read or write position in the LOB stream in bytes
Value	Returns the data, starting from the first character in the `CLOB` or `NCLOB`, as a string

OracleClob Instance Methods

The `OracleClob` instance methods are listed in Table 5–169.

Table 5–169 OracleClob Instance Methods

Methods	Description
Append	Appends data to the current `OracleClob` instance (Overloaded)
BeginChunkWrite	Opens the `CLOB`
`BeginRead`	Inherited from `Stream`
`BeginWrite`	Inherited from `Stream`
Clone	Creates a copy of an `OracleClob` object
Close	Closes the current stream and releases resources associated with it
Compare	Compares data referenced by the current instance to that of the supplied object
CopyTo	Copies the data to an `OracleClob` (Overloaded)
`CreateObjRef`	Inherited from `MarshalByRefObject`
Dispose	Releases resources allocated by this object

Table 5–169 OracleClob Instance Methods (Cont.)

Methods	Description
EndChunkWrite	Closes the `CLOB` referenced by the current `OracleClob` instance
`EndRead`	Inherited from `Stream`
`EndWrite`	Inherited from `Stream`
`Equals`	Inherited from `Object` (Overloaded)
Erase	Erases the specified `amount` of data (Overloaded)
`Flush`	*Not supported*
GetHashCode	Returns a hash code for the current instance
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
IsEqual	Compares the LOB data referenced by two `OracleClobs`
Read	Reads from the current instance (Overloaded)
`ReadByte`	Inherited from `Stream`
Search	Searches for a character pattern in the current instance of `OracleClob` (Overloaded)
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the `CLOB` value
`ToString`	Inherited from `Object`
Write	Writes the provided `buffer` into the `OracleClob` (Overloaded)
`WriteByte`	Inherited from `Stream`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class

OracleClob Constructors

`OracleClob` constructors create instances of the `OracleClob` class bound to a temporary `CLOB`.

Overload List:

- OracleClob(OracleConnection)

  This constructor creates an instance of the `OracleClob` class bound to a temporary `CLOB` with an `OracleConnection` object.

- OracleClob(OracleConnection, bool, bool)

  This constructor creates an instance of the `OracleClob` class that is bound to a temporary `CLOB`, with an `OracleConnection` object, a boolean value for caching, and a boolean value for `NCLOB`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

OracleClob(OracleConnection)

This constructor creates an instance of the `OracleClob` class bound to a temporary `CLOB` with an `OracleConnection` object.

Declaration

```
// C#
public OracleClob(OracleConnection con);
```

Parameters

- *con*

  The `OracleConnection` object.

Exceptions

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The connection must be opened explicitly by the application. `OracleClob` does not open the connection implicitly. The temporary `CLOB` utilizes the provided connection to store `CLOB` data. Caching is not enabled by default.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

OracleClob(OracleConnection, bool, bool)

This constructor creates an instance of the `OracleClob` class that is bound to a temporary `CLOB`, with an `OracleConnection` object, a boolean value for caching, and a boolean value for `NCLOB`.

Declaration

```
// C#
public OracleClob(OracleConnection con, bool bCaching, bool bNCLOB);
```

Parameters

- *con*

  The `OracleConnection` object connection.

- *bCaching*

  A flag that indicates whether or not server-side caching is enabled.

- *bNCLOB*

  A flag that is set to `true` if the instance is a `NCLOB` or `false` if it is a `CLOB`.

Exceptions

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The connection must be opened explicitly by the application. `OracleClob` does not open the connection implicitly. The temporary `CLOB` or `NCLOB` uses the provided connection to store `CLOB` data.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

OracleClob Static Fields

`OracleClob` static fields are listed in Table 5–170.

Table 5–170 OracleClob Static Fields

Field	Description
MaxSize	Holds the maximum number of bytes a `CLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

MaxSize

The `MaxSize` field holds the maximum number of bytes a `CLOB` can hold, which is 4,294,967,295 (2^32 - 1) bytes.

Declaration

```
// C#
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field is useful in code that checks whether your operation exceeds the maximum length (in bytes) allowed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

OracleClob Static Methods

`OracleClob` static methods are listed in Table 5–171.

Table 5–171 OracleClob Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

OracleClob Instance Properties

`OracleClob` instance properties are listed in Table 5–172.

Table 5–172 OracleClob Instance Properties

Properties	Description
CanRead	Indicates whether the LOB stream can be read
CanSeek	Indicates whether forward and backward seek operations can be performed
CanWrite	Indicates whether the LOB stream can be written
Connection	Indicates the `OracleConnection` that is used to retrieve and write `CLOB` data
IsEmpty	Indicates whether the `CLOB` is `empty` or not
IsInChunkWriteMode	Indicates whether or not the `CLOB` has been opened
IsNCLOB	Indicates whether the `OracleClob` object represents an `NCLOB`.
IsTemporary	Indicates whether or not the current instance is bound to a temporary `CLOB`
Length	Indicates the size of the `CLOB` data in bytes
OptimumChunkSize	Indicates the minimum number of bytes to retrieve or send from the server during a read or write operation
Position	Indicates the current read or write position in the LOB stream in bytes

Table 5–172 OracleClob Instance Properties (Cont.)

Properties	Description
Value	Returns the data, starting from the first character in the `CLOB` or `NCLOB`, as a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

CanRead

Overrides `Stream`

This instance property indicates whether the LOB stream can be read.

Declaration

```
// C#
public override bool CanRead{get;}
```

Property Value

If the LOB stream can be read, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

CanSeek

Overrides `Stream`

This instance property indicates whether forward and backward seek operations can be performed.

Declaration

```
// C#
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

CanWrite

Overrides `Stream`

This instance property indicates whether the LOB object supports writing.

Declaration

```
// C#
public override bool CanWrite{get;}
```

Property Value

If the LOB stream can be written, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Connection

This instance property indicates the `OracleConnection` that is used to retrieve and write `CLOB` data.

Declaration

```
// C#
public OracleConnection Connection {get;}
```

Property Value

An `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

IsEmpty

This instance property indicates whether the `CLOB` is empty or not.

Declaration

```
// C#
public bool IsEmpty {get;}
```

Property Value

A `bool`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

IsInChunkWriteMode

This instance property indicates whether the `CLOB` has been opened to defer index updates.

Declaration

```
// C#
public bool IsInChunkWriteMode{get;}
```

Property Value

If the `CLOB` has been opened, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

IsNCLOB

This instance property indicates whether the `OracleClob` object represents an `NCLOB`.

Declaration

```
// C#
public bool IsNCLOB {get;}
```

Property Value

A `bool`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

IsTemporary

This instance property indicates whether or not the current instance is bound to a temporary `CLOB`.

Declaration

```
// C#
public bool IsTemporary {get;}
```

Property Value

A `bool`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Length

Overrides `Stream`

This instance property indicates the size of the `CLOB` data in bytes.

Declaration

```
// C#
public override Int64 Length {get;}
```

Property Value

An `Int64` that indicates the size of the `CLOB` in bytes.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

OptimumChunkSize

This instance property indicates the minimum number of bytes to retrieve or send from the server during a read or write operation.

Declaration

```
// C#
public int OptimumChunkSize{get;}
```

Property Value

A number representing the minimum bytes to retrieve or send.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Position

Overrides `Stream`

This instance property indicates the current read or write position in the LOB stream in bytes.

Declaration

```
// C#
public override Int64 Position{get; set;}
```

Property Value

An `Int64` that indicates the read or write position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The `Position` is less than `0`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Value

This instance property returns the data, starting from the first character in the CLOB or NCLOB, as a string.

Declaration

```
// C#
public string Value{get;}
```

Property Value

A string.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The Value is less than 0.

Remarks

The value of Position is neither used nor changed by using this property.

The maximum string length that can be returned by this property is 2 GB.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

OracleClob Instance Methods

The OracleClob instance methods are listed in Table 5–173.

Table 5–173 OracleClob Instance Methods

Methods	Description
Append	Appends data to the current OracleClob instance (Overloaded)
BeginChunkWrite	Opens the CLOB
BeginRead	Inherited from Stream

Table 5–173 OracleClob Instance Methods (Cont.)

Methods	Description
BeginWrite	Inherited from Stream
Clone	Creates a copy of an OracleClob object
Close	Closes the current stream and releases resources associated with it
Compare	Compares data referenced by the current instance to that of the supplied object
CopyTo	Copies the data to an OracleClob (Overloaded)
CreateObjRef	Inherited from MarshalByRefObject
Dispose	Releases resources allocated by this object
EndChunkWrite	Closes the CLOB referenced by the current OracleClob instance
EndRead	Inherited from Stream
EndWrite	Inherited from Stream
Equals	Inherited from Object (Overloaded)
Erase	Erases the specified amount of data (Overloaded)
Flush	*Not supported*
GetHashCode	Returns a hash code for the current instance
GetLifetimeService	Inherited from MarshalByRefObject
GetType	Inherited from Object
InitializeLifetimeService	Inherited from MarshalByRefObject
IsEqual	Compares the LOB data referenced by two OracleClobs
Read	Reads from the current instance (Overloaded)
ReadByte	Inherited from Stream
Search	Searches for a character pattern in the current instance of OracleClob (Overloaded)
Seek	Sets the position in the current LOB stream
SetLength	Trims or truncates the CLOB value
ToString	Inherited from Object

Table 5–173 OracleClob Instance Methods (Cont.)

Methods	Description
Write	Writes the provided `buffer` into the `OracleClob` (Overloaded)
`WriteByte`	Inherited from `Stream`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Append

This instance method appends data to the current `OracleClob` instance.

Overload List:

- Append(OracleClob)

  This instance method appends the `CLOB` data referenced by the provided `OracleClob` object to the current `OracleClob` instance.

- Append(byte [ ], int, int)

  This instance method appends data at the end of the `CLOB`, from the supplied byte array buffer, starting from offset (in bytes) of the supplied byte array buffer.

- Append(char [ ], int, int)

  This instance method appends data from the supplied character array buffer to the end of the current `OracleClob` instance, starting at the offset (in characters) of the supplied character buffer.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Append(OracleClob)

This instance method appends the CLOB data referenced by the provided OracleClob object to the current OracleClob instance.

Declaration

```
// C#
public void Append(OracleClob obj);
```

Parameters

- *obj*

  An OracleClob object.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The parameter has a different connection than the object, OracleConnection is not opened, or OracleConnection has been reopened.

Remarks

No character set conversions are made.

The provided object and the current instance must be using the same connection; that is, the same OracleConnection object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Append(byte [ ], int, int)

This instance method appends data at the end of the CLOB, from the supplied byte array buffer, starting from offset (in bytes) of the supplied byte array buffer.

Declaration

```
// C#
public int Append(byte[] buffer, int offset, int count);
```

Parameters

- *buffer*

  An array of bytes, representing a Unicode string.

- *offset*

  The zero-based byte offset in the buffer from which data is read.

- *count*

  The number of bytes to be appended.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - Either the *offset* or the *count* parameter is not even.

Remarks

Both *offset* and *count* must be even numbers for `CLOB` and `NCLOB` because every two bytes represent a Unicode character.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Append(char [ ], int, int)

This instance method appends data from the supplied character array buffer to the end of the current `OracleClob` instance, starting at the offset (in characters) of the supplied character buffer.

Declaration

```
// C#
public void Append(char[] buffer, int offset, int count);
```

Parameters

- *buffer*

An array of characters.

- *offset*

  The zero-based offset (in characters) in the buffer from which data is read.

- *count*

  The number of characters to be appended.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Example

```
// C#
...
// Append 2 char to the oraClob
oraClob.Append(new char[3] {"f", "o", "o"}, 1, 2);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

BeginChunkWrite

This instance method opens the `CLOB`.

Declaration

```
// C#
public void BeginChunkWrite();
```

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

`BeginChunkWrite` does not need to be called before manipulating the `CLOB` data. This is provided for performance reasons.

After this method is called, write operations do not cause the domain or function-based index on the column to be updated. Index updates occur only once after `EndChunkWrite` is called.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Clone

This instance method creates a copy of an `OracleClob` object.

Declaration

```
// C#
public object Clone();
```

Return Value

An `OracleClob` object.

Implements

`ICloneable`

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
//Need a proper casting for the return value when cloned
```

```
OracleClob oraClob_cloned = (OracleClob) oraClob.Clone();
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Close

Overrides `Stream`

This instance method closes the current stream and releases resources associated with it.

Declaration

```
// C#
public override void Close();
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Compare

This instance method compares data referenced by the current instance to that of the supplied object.

Declaration

```
// C#
public int Compare(Int64 src_offset, OracleClob obj, Int64 dst_offset,
  Int64 amount);
```

Parameters

- *src_offset*

  The comparison starting point (in characters) for the current instance.

- *obj*

The provided `OracleClob` object.

- *`dst_offset`*

  The comparison starting point (in characters) for the provided `OracleClob`.

- *`amount`*

  The number of characters to compare.

Return Value

The method returns a value that is:

- Less than zero: if the data referenced by the current instance is less than that of the supplied instance.
- Zero: if both objects reference the same data.
- Greater than zero: if the data referenced by the current instance is greater than that of the supplied instance.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - Either the *`src_offset`*, *`dst_offset`*, or *`amount`* parameter is less than `0`.

Remarks

The character set of the two `OracleClob` objects being compared should be the same for a meaningful comparison.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

CopyTo

`CopyTo` copies data from the current instance to the provided `OracleClob` object.

Overload List:

- CopyTo(OracleClob)

  This instance method copies data from the current instance to the provided `OracleClob` object.

- CopyTo(OracleClob, Int64)

  This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified destination offset.

- CopyTo(Int64, OracleClob, Int64, Int64)

  This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified source offset, destination offset, and character amounts.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

CopyTo(OracleClob)

This instance method copies data from the current instance to the provided `OracleClob` object.

Declaration

```
// C#
public Int64 CopyTo(OracleClob obj);
```

Parameters

- *obj*

  The `OracleClob` object to which the data is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

CopyTo(OracleClob, Int64)

This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified destination offset.

Declaration

```
// C#
public Int64 CopyTo(OracleClob obj, Int64 dst_offset);
```

Parameters

- *obj*

  The `OracleClob` object to which the data is copied.

- *dst_offset*

  The offset (in characters) at which the `OracleClob` object is copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentOutOfRangeException` - The *`dst_offset`* is less than `0`.

`InvalidOperationException` - This exception is thrown if any of the following conditions exist:

- The `OracleConnection` is not open or has been closed during the lifetime of the object.
- The LOB object parameter has a different connection than the object.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleClob` data, spaces are written into the `OracleClob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

CopyTo(Int64, OracleClob, Int64, Int64)

This instance method copies data from the current `OracleClob` instance to the provided `OracleClob` object with the specified source offset, destination offset, and character amounts.

Declaration

```
// C#
public Int64 CopyTo(Int64 src_offset,OracleClob obj,Int64 dst_offset,Int64
amount);
```

Parameters

- *`src_offset`*

  The offset (in characters) in the current instance, from which the data is read.

- *`obj`*

The `OracleClob` object to which the data is copied.

- *`dst_offset`*

  The offset (in characters) at which the `OracleClob` object is copied.

- *`amount`*

  The amount of data to be copied.

Return Value

The return value is the amount copied.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

`ArgumentOutOfRangeException` - The *`src_offset`*, the *`dst_offset`*, or the *`amount`* parameter is less than `0`.

Remarks

If the *`dst_offset`* is beyond the end of the `OracleClob` data, spaces are written into the `OracleClob` until the *`dst_offset`* is met.

The offsets are `0`-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

Example

```
// C#
...
// Assume you have a valid connection 'con'
OracleClob src_clob = new OracleClob(con);
OracleClob target_clob = new OracleClob(con);

// Copy 1024 chars from src_clob (begin at offset 10) to target_blob
// (starting at offset 5)
src_clob.CopyTo(10, target_clob, 5, 1024);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Dispose

This instance method releases resources allocated by this object.

Declaration

```
public void Dispose();
```

Implements

`IDisposable`

Remarks

The object cannot be reused after being disposed. Although some properties can still be accessed, their values cannot be accountable. Since resources are freed, method calls can lead to exceptions.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

EndChunkWrite

This instance method closes the `CLOB` referenced by the current `OracleClob` instance.

Declaration

```
// C#
public void EndChunkWrite();
```

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

Index updates occur immediately if write operation(s) are deferred by the `BeginChunkWrite` method.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Erase

`Erase` erases part or all data.

Overload List:

- Erase()

  This instance method erases all data.

- Erase(Int64, Int64)

  This instance method replaces the specified `amount` of data (in characters) starting from the specified `offset` with zero-byte fillers (in characters).

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Erase()

This instance method erases all data.

Declaration

```
// C#
public Int64 Erase();
```

Return Value

The number of characters erased.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Erase(Int64, Int64)

This instance method replaces the specified `amount` of data (in characters) starting from the specified `offset` with zero-byte fillers (in characters).

Declaration

```
// C#
public Int64 Erase(Int64 offset, Int64 amount);
```

Parameters

- *`offset`*

  The offset.

- *`amount`*

  The amount of data.

Return Value

The actual number of characters erased.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The *`offset`* or *`amount`* parameter is less than `0`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

GetHashCode

Overrides `Object`

This method returns a hash code for the current instance.

Declaration

```
// C#
public override int GetHashCode();
```

Return Value

An `int` representing a hash code.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

IsEqual

This instance method compares the LOB data referenced by two `OracleClobs`.

Declaration

```
// C#
public bool IsEqual(OracleClob obj);
```

Parameters

- *obj*

  An `OracleClob` object.

Return Value

Returns `true` if the current `OracleClob` and the provided `OracleClob` refer to the same LOB. Otherwise, returns `false`.

Remarks

Note that this method can return `true` even if the two `OracleClob` objects returns `false` for == or `Equals()` because two different `OracleClob` instances can refer to the same LOB.

The provided object and the current instance must be using the same connection, that is, the same `OracleConnection` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Read

`Read` reads a specified amount from the current instance and populates the array buffer.

Overload List:

- Read(byte [ ], int, int)

  This instance method reads a specified amount of bytes from the current instance and populates the byte array `buffer`.

- Read(char [ ], int, int)

  This instance method reads a specified amount of characters from the current instance and populates the character array buffer.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Read(byte [ ], int, int)

Overrides `Stream`

This instance method reads a specified amount of bytes from the current instance and populates the byte array `buffer`.

Declaration

```
// C#
public override int Read(byte [ ] buffer, int offset, int count);
```

Parameters

- *buffer*

  The byte array buffer that is populated.

- *offset*

  The offset (in bytes) at which the buffer is populated.

- *count*

  The amount of bytes to be read.

Return Value

The number of bytes read from the CLOB.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

Both *offset* and *count* must be even numbers for CLOB and NCLOB because every two bytes represent a Unicode character.

The LOB data is read starting from the position specified by the Position property, which must also be an even number.

OracleClob is free to return fewer bytes than requested, even if the end of the stream has not been reached.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Read(char [ ], int, int)

This instance method reads a specified amount of characters from the current instance and populates the character array buffer.

Declaration

```
// C#
public int Read(char[ ] buffer, int offset, int count);
```

Parameters

- *buffer*

  The character array buffer that is populated.

- *offset*

  The offset (in characters) at which the buffer is populated.

- *count*

  The amount of characters to be read.

Return Value

The return value indicates the number of characters read from the CLOB.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The *offset* or the *count* is less than 0.
- The *offset* is greater than or equal to the *buffer*.Length.
- The *offset* and the *count* together are greater than *buffer*.Length.

Remarks

Handles all CLOB and NCLOB data as Unicode.

The LOB data is read starting from the position specified by the Position property.

Example

```
// C#
...
// Read some data (in characters)
```

```
char buffer = new char[1024];
int  bufferOffset  = 10;
int amountToBeRead = 10;

int charRead = oraClob.Read(buffer, bufferOffset, amountToBeRead);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Search

`Search` searches for a character pattern in the current instance of `OracleClob`.

Overload List:

- Search(byte[ ], Int64, Int64)

  This instance method searches for a character pattern, represented by the byte array, in the current instance of `OracleClob`.

- Search(char[ ], Int64, Int64)

  This instance method searches for a character pattern in the current instance of `OracleClob`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Search(byte[ ], Int64, Int64)

This instance method searches for a character pattern, represented by the byte array, in the current instance of `OracleClob`.

Declaration

```
// C#
public int Search(byte[ ] val, Int64 offset, Int64 nth);
```

Parameters

- *val*

  A Unicode byte array.

- *offset*

  The 0-based offset (in characters) starting from which the OracleClob is searched.

- *nth*

  The specific occurrence (1-based) of the match for which the absolute offset (in characters) is returned.

Return Value

Returns the absolute *offset* of the start of the matched pattern (in bytes) for the *nth* occurrence of the match. Otherwise, 0 is returned.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The *offset* is less than 0.
- The *nth* is less than or equal to 0.
- The *nth* is greater than or equal to OracleClob.MaxSize.
- The *offset* is greater than or equal to OracleClob.MaxSize.

Remarks

The byte[] is converted to Unicode before the search is made.

The limit of the search pattern is 16383 bytes.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Search(char[ ], Int64, Int64)

This instance method searches for a character pattern in the current instance of `OracleClob`.

Declaration

```
// C#
public Int64 Search(char [ ] val, Int64 offset, Int64 nth);
```

Parameters

- *`val`*

  The Unicode string being searched for.

- *`offset`*

  The `0`-based offset (in characters) starting from which the `OracleClob` is searched.

- *`nth`*

  The specific occurrence (`1`-based) of the match for which the absolute offset (in characters) is returned.

Return Value

Returns the absolute *`offset`* of the start of the matched pattern (in characters) for the *`nth`* occurrence of the match. Otherwise, `0` is returned.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *`offset`* is less than `0`.
- The *`nth`* is less than or equal to `0`.
- The *`val`*`.Length` doubled is greater than `16383`.
- The *`nth`* is greater than or equal to `OracleClob.MaxSize`.
- The *`offset`* is greater than or equal to `OracleClob.MaxSize`.

Remarks

The limit of the search pattern is 16383 bytes.

Examples

```
// C#
..
// Search for the 2nd occurrence of a char pattern 'oracle'
// from the oraClob starting at offset 1
char[6] pattern = new char[6] { "o", "r", "a", "c", "l", "e" };
int positionFound = oraClob.Search(pattern, 1, 2);
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Seek

Overrides `Stream`

This instance method sets the position on the current LOB stream.

Declaration

```
// C#
public override Int64(Int64 offset, SeekOrigin origin);
```

Parameters

- *offset*

  A byte offset relative to origin.

- *origin*

  A value of type `System.IO.SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

Returns an `Int64` that indicates the position.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

If *`offset`* is negative, the new position precedes the position specified by *`origin`* by the number of characters specified by *`offset`*.

If *`offset`* is zero, the new position is the position specified by *`origin`*.

If *`offset`* is positive, the new position follows the position specified by *`origin`* by the number of characters specified by *`offset`*.

`SeekOrigin.Begin` specifies the beginning of a stream.

`SeekOrigin.Current` specifies the current position within a stream.

`SeekOrigin.End` specifies the end of a stream.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

SetLength

Overrides `Stream`

This instance method trims or truncates the `CLOB` value to the specified length (in characters).

Declaration

```
// C#
public override void SetLength(Int64 newlen);
```

Parameters

- *`newlen`*

  The desired length of the current stream in characters.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The *`newlen`* parameter is greater than `0`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Write

This instance method writes data from the provided array buffer into the `OracleClob`.

Overload List:

- Write(byte[ ], int, int)

  This instance method writes data from the provided byte array buffer into the `OracleClob`.

- Write(char[ ], int, int)

  This instance method writes data from the provided character array buffer into the `OracleClob`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Write(byte[ ], int, int)

Overrides `Stream`

This instance method writes data from the provided byte array buffer into the `OracleClob`.

Declaration

```
// C#
public override void Write(byte[ ] buffer, int offset, int count);
```

Parameters

- *buffer*

  The byte array buffer that represents a Unicode string.

- *offset*

  The offset (in bytes) from which the buffer is read.

- *count*

  The amount of data (in bytes) from the buffer to be written into the OracleClob.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The *offset* or the *count* is less than 0.
- The *offset* is greater than or equal to the *buffer*.Length.
- The *offset* and the *count* together are greater than the *buffer*.Length.
- The *offset*, the *count*, or the Position is not even.

Remarks

Both *offset* and *count* must be even numbers for CLOB and NCLOB because every two bytes represent a Unicode character.

The LOB data is read starting from the position specified by the Position property. The Position property must be an even number.

If necessary, proper data conversion is carried out from the client character set to the database character set.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

Write(char[ ], int, int)

This instance method writes data from the provided character array buffer into the `OracleClob`.

Declaration

```
// C#
public void Write(char[ ] buffer, int offset, int count);
```

Parameters

- *buffer*

  The character array buffer that is written to the `OracleClob`.

- *offset*

  The offset (in characters) from which the *buffer* is read.

- *count*

  The amount (in characters) from the buffer that is to be written into the `OracleClob`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - This exception is thrown if any of the following conditions exist:

- The *offset* or the *count* is less than `0`.
- The *offset* is greater than or equal to the *buffer*.`Length`.
- The *offset* and the *count* together are greater than *buffer*.`Length`.
- The `Position` is not even.

Remarks

Handles all `CLOB` and `NCLOB` data as Unicode.

The LOB data is read starting from the position specified by the `Position` property.

If necessary, proper data conversion is carried out from the client character set to the database character set.

Example

```
// C#
...
// Begin ChunkWrite to improve performance
// Index updates occur only once after EndChunkWrite
oraClob.BeginChunkWrite();

// Set the write from the beginning;
oraClob.Position = 0;

// Write to the oraClob in chunks of 10, each 1024 char
for ( int i=0; i<10; i++ )
 {
   char[1024] c;
   c = c[0]= a;    // some new value to be written
   oraClob.Write(c, 0, c.Length);
 }

oraClob.EndChunkWrite();
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleClob Class
- OracleClob Members

OracleRefCursor Class

An `OracleRefCursor` object represents an Oracle `REF CURSOR`.

Class Inheritance

```
Object
  MarshalRefByObject
    OracleRefCursor
```

Declaration

```
// C#
public sealed class OracleRefCursor : MarshalRefByObject, IDisposable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Example

```
// C#
// Example demonstrates how to use a REF CURSOR as an output parameter
...
// 1. Assume you have a stored procedure called MyPack.MyProc created with
// the following function, which returns 1 REF CURSOR and contains one REF
// CURSOR as an output parameter
//    FUNCTION MyProc (...)
//    BEGIN
//      open p_cursor for select * from multimedia_tab;
//      open p_cursor1 for select * from emp;
//      return (p_cursor);
//    END MyProc;
//    ...
//
// 2. Assume you have a valid connection
// Set the command
OracleCommand cmd = new OracleCommand("MyPack.MyProc", con);
cmd.CommandType = CommandType.StoredProcedure;

// Bind
// select * from multimedia_tab;
OracleParameter p1 = cmd.Parameters.Add("refcursor1", OracleDbType.RefCursor);
```

```
p1.Direction = ParameterDirection.ReturnValue;

// select * from emp
OracleParameter p2 = cmd.Parameters.Add("refcursor2", OracleDbType.RefCursor);
p2.Direction = ParameterDirection.Output;

// Execute command
cmd.ExecuteNonQuery();
...
DataReader reader = p1.Value.GetDataReader();
Console.WriteLine("Field count: " + reader.FieldCount);
...
```

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleRefCursor Members
- OracleRefCursor Static Methods
- OracleRefCursor Properties
- OracleRefCursor Instance Methods

OracleRefCursor Members

`OracleRefCursor` members are listed in the following tables:

OracleRefCursor Static Methods

`OracleRefCursor` static methods are listed in Table 5–174.

Table 5–174 OracleRefCursor Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleRefCursor Properties

`OracleRefCursor` properties are listed in Table 5–175.

Table 5–175 OracleRefCursor Properties

Properties	Description
Connection	A reference to the `OracleConnection` used to fetch the `REF CURSOR` data

OracleRefCursor Instance Methods

`OracleRefCursor` instance methods are listed in Table 5–176.

Table 5–176 OracleRefCursor Instance Methods

Methods	Description
Dispose	Disposes the resources allocated by the `OracleRefCursor` object
`Equals`	Inherited from `Object` (Overloaded)
GetDataReader	Returns an `OracleDataReader` object for the `REF CURSOR`
`GetHashCode`	Inherited from `Object`
`GetType`	Inherited from `Object`
`ToString`	Inherited from `Object`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleRefCursor Class

OracleRefCursor Static Methods

`OracleRefCursor` static methods are listed in Table 5–177.

Table 5–177 OracleRefCursor Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleRefCursor Class
- OracleRefCursor Members

OracleRefCursor Properties

`OracleRefCursor` properties are listed in Table 5–178.

Table 5–178 OracleRefCursor Properties

Properties	Description
Connection	A reference to the `OracleConnection` used to fetch the `REF CURSOR` data

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleRefCursor Class
- OracleRefCursor Members

Connection

This property refers to the `OracleConnection` used to fetch the `REF CURSOR` data.

Declaration

```
// C#
public OracleConnection Connection {get;}
```

Property Value

An `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

This property is bound to a `REF CURSOR` once it is set. After the `OracleRefCursor` object is created by the constructor, this property is initially

null. An OracleRefCursor object can be bound to a REF CURSOR after a command execution.

If the connection is closed or returned to the connection pool, the OracleRefCursor is placed in an uninitialized state and no operation can be carried out from it. However, the uninitialized OracleRefCursor can be reassigned to another REF CURSOR.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleRefCursor Class
- OracleRefCursor Members

OracleRefCursor Instance Methods

OracleRefCursor instance methods are listed in Table 5–179.

Table 5–179 OracleRefCursor Instance Methods

Methods	Description
Dispose	Disposes the resources allocated by the OracleRefCursor object
Equals	Inherited from Object (Overloaded)
GetDataReader	Returns an OracleDataReader object for the REF CURSOR
GetHashCode	Inherited from Object
GetType	Inherited from Object
ToString	Inherited from Object

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleRefCursor Class
- OracleRefCursor Members

Dispose

This instance method disposes of the resources allocated by the OracleRefCursor object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

Remarks

The object cannot be reused after being disposed.

Once `Dispose()` is called, the object of `OracleRefCursor` is in an uninitialized state. Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls can lead to exceptions.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleRefCursor Class
- OracleRefCursor Members

GetDataReader

This instance method returns an `OracleDataReader` object for the `REF CURSOR`.

Declaration

```
// C#
public OracleDataReader GetDataReader();
```

Return Value

`OracleDataReader`

Remarks

Using the `OracleDataReader`, rows can be fetched from the `REF CURSOR`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleRefCursor Class
- OracleRefCursor Members

OracleXmlStream Class

An `OracleXmlStream` object represents a read-only stream of XML data stored in an `OracleXmlType` object.

Class Inheritance

```
Object
  MarshalByRefObject
    Stream
      OracleXmlStream
```

Declaration

```
// C#
public sealed class OracleXmlStream : IDisposable, ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

This class can only be used with Oracle9*i* Release 2 (9.2) and later.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Members
- OracleXmlStream Constructor
- OracleXmlStream Static Methods
- OracleXmlStream Instance Properties
- OracleXmlStream Instance Methods

OracleXmlStream Members

`OracleXmlStream` members are listed in the following tables:

OracleXmlStream Constructors

The `OracleXmlStream` constructors are listed in Table 5–180.

Table 5–180 OracleXmlStream Constructors

Constructor	Description
OracleXmlStream Constructor	Creates an instance of an `OracleXmlStream` object which provides a `Stream` representation of the XML data stored in an `OracleXmlType`

OracleXmlStream Static Methods

The `OracleXmlStream` static methods are listed in Table 5–181.

Table 5–181 OracleXmlStream Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleXmlStream Instance Properties

The `OracleXmlStream` instance properties are listed in Table 5–182.

Table 5–182 OracleXmlStream Instance Properties

Properties	Description
CanRead	Indicates whether the XML stream can be read
CanSeek	Indicates whether forward and backward seek operation can be performed
CanWrite	*Not Supported*
Connection	Indicates the `OracleConnection` that is used to retrieve the XML data
Length	Indicates the number of bytes in the XML stream
Position	Gets or sets the byte position within the stream
Value	Returns the XML data, starting from the first character in the stream as a string

OracleXmlStream Instance Methods

The `OracleXmlStream` instance methods are listed in Table 5–183.

Table 5–183 OracleXmlStream instance Methods

Methods	Description
`BeginRead`	Inherited from `Stream`
`BeginWrite`	Inherited from `Stream`
Clone	Creates a copy of an `OracleXmlStream` object
Close	Closes the current stream and releases any resources associated with it
Dispose	Releases resources allocated by this object
`EndRead`	Inherited from `Stream`
`EndWrite`	Inherited from `Stream`
`Equals`	Inherited from `Object`
`Flush`	*Not Supported*
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`
`InitializeLifetimeService`	Inherited from `MarshalByRefObject`
Read	Reads a specified amount from the current stream instance and populates the array buffer (Overloaded)
`ReadByte`	Inherited from `Stream`
Seek	Sets the position within the current stream and returns the new position within the current stream
`SetLength`	*Not Supported*
`ToString`	Inherited from `Object`
`Write`	*Not Supported*
`WriteByte`	*Not Supported*

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class

OracleXmlStream Constructor

This constructor creates an instance of an `OracleXmlStream` object which provides a `Stream` representation of the XML data stored in an `OracleXmlType` object.

Declaration

```
// C#
public OracleXmlStream(OracleXmlType xmlType);
```

Parameters

- *xmlType*

  The `OracleXmlType` object.

Remarks

The `OracleXmlStream` implicitly uses the `OracleConnection` object from the `OracleXmlType` object from which it was constructed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

OracleXmlStream Static Methods

The `OracleXmlStream` static methods are listed in Table 5–184.

Table 5–184 OracleXmlStream Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

OracleXmlStream Instance Properties

The `OracleXmlStream` instance properties are listed in Table 5–185.

Table 5–185 OracleXmlStream Instance Properties

Properties	Description
CanRead	Indicates whether the XML stream can be read
CanSeek	Indicates whether forward and backward seek operation can be performed
`CanWrite`	*Not Supported*
Connection	Indicates the `OracleConnection` that is used to retrieve the XML data
Length	Indicates the number of bytes in the XML stream
Position	Gets or sets the byte position within the stream
Value	Returns the XML data, starting from the first character in the stream as a string

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

CanRead

Overrides `Stream`

This property indicates whether the XML stream can be read.

Declaration

```
// C#
public override bool CanRead{get;}
```

Property Value

If the XML stream is can be read, returns `true`; otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

CanSeek

Overrides `Stream`

This property indicates whether forward and backward seek operation can be performed.

Declaration

```
// C#
public override bool CanSeek{get;}
```

Property Value

If forward and backward seek operations can be performed, this property returns `true`. Otherwise, returns `false`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Connection

This instance property indicates the `OracleConnection` that is used to retrieve the XML data.

Declaration

```
// C#
public OracleConnection Connection {get;}
```

Property Value

An `OracleConnection`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Length

Overrides `Stream`

This property indicates the number of bytes in the XML stream.

Declaration

```
// C#
public override Int64 Length{get;}
```

Property Value

An `Int64` value representing the number of bytes in the XML stream. An empty stream has a length of 0 bytes.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Position

Overrides `Stream`

This property gets or sets the byte position within the stream.

Declaration

```
// C#
public override Int64 Position{get; set;}
```

Property Value

An `Int64` that indicates the current position in the stream.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentOutOfRangeException` - The `Position` is less than 0.

Remarks

The beginning of the stream is represented by position `0`. Seeking to any location beyond the length of the stream is supported.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Value

This property returns the XML data, starting from the first character of the stream as a string.

Declaration

```
// C#
public string Value{get; set;}
```

Property Value

A `string`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The value of `Position` is neither used nor changed by using this property.

The maximum length of the string that can be returned by this property is 2 GB.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

OracleXmlStream Instance Methods

The `OracleXmlStream` instance methods are listed in Table 5–186.

Table 5–186 OracleXmlStream Instance Methods

Methods	Description
`BeginRead`	Inherited from `Stream`
`BeginWrite`	Inherited from `Stream`
Clone	Creates a copy of an `OracleXmlStream` object
Close	Closes the current stream and releases any resources associated with it
Dispose	Releases resources allocated by this object
`EndRead`	Inherited from `Stream`
`EndWrite`	Inherited from `Stream`
`Equals`	Inherited from `Object`
`Flush`	*Not Supported*
`GetHashCode`	Inherited from `Object`
`GetLifetimeService`	Inherited from `MarshalByRefObject`
`GetType`	Inherited from `Object`

Table 5–186 OracleXmlStream Instance Methods (Cont.)

Methods	Description
InitializeLifetimeService	Inherited from MarshalByRefObject
Read	Reads a specified amount from the current XML stream instance and populates the array buffer (Overloaded)
ReadByte	Inherited from Stream
Seek	Sets the position within the current stream and returns the new position within the current stream
SetLength	*Not Supported*
ToString	Inherited from Object
Write	*Not Supported*
WriteByte	*Not Supported*

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Clone

This method creates a copy of an OracleXmlStream object.

Declaration

```
// C#
public object Clone();
```

Return Value

An OracleXmlStream object.

Implements

ICloneable

Exceptions

ObjectDisposedException - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The cloned object has the same property values as that of the object being cloned.

Example

```
// C#
...
//Need a proper casting for the return value when cloned
OracleXmlStream oraXmlStream_cloned = (OracleXmlStream) oraXmlStream.Clone();
...
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Close

Overrides `Stream`

This method closes the current stream and releases any resources associated with it.

Declaration

```
// C#
public override void Close();
```

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Dispose

This public method releases resources allocated by this object.

Declaration

```
// C#
```

```
public void Dispose();
```

Implements

`IDisposable`

Remarks

The object cannot be reused after being disposed. Although some properties can still be accessed, their values cannot be accountable. Since resources are freed, method calls can lead to exceptions.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Read

This method reads a specified amount from the current XML stream instance and populates the array buffer.

Overload List:

- Read(byte[ ], int, int)

  This method reads a specified amount of unicode bytes from the current instance, advances the position within the stream, and populates the byte array buffer.

- Read(char[ ], int, int)

  This method reads a specified amount of characters from the current instance, advances the position within the stream, and populates the character array buffer.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Read(byte[ ], int, int)

Overrides `Stream`

This method reads a specified amount of unicode bytes from the current instance, advances the position within the stream, and populates the byte array buffer.

Declaration

```
// C#
public override int Read(byte[ ] buffer, int offset, int count);
```

Parameters

- *buffer*

  The byte array buffer that is populated.

- *offset*

  The zero-based offset (in bytes) at which the buffer is populated.

- *count*

  The maximum amount of bytes to be read.

Return Value

The number of unicode bytes read into the given `byte[]` buffer or `0` if the end of the stream has been reached.

Remarks

This method reads a maximum of *count* bytes from the current stream and stores them in buffer beginning at *offset*. The current position within the stream is advanced by the number of bytes read. However, if an exception occurs, the current position within the stream remains unchanged.

The XML data is read starting from the position specified by the `Position` property.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Read(char[ ], int, int)

Overrides `Stream`

This method reads a specified amount of characters from the current instance, advances the position within the stream, and populates the character array buffer.

Declaration

```
// C#
public override int Read(char[ ] buffer, int offset, int count);
```

Parameters

- *`buffer`*

  The character array buffer to be populated.

- *`offset`*

  The zero-based offset (in characters) in the buffer at which the buffer is populated.

- *`count`*

  The maximum amount of characters to be read from the stream.

Return Value

The return value indicates the number of characters read from the stream or `0` if the end of the stream has been reached.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

This method requires that the `Position` on the stream instance be zero or an even number.

The XML data is read starting from the position specified by the `Position` property.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

Seek

Overrides `Stream`.

This method sets the position within the current stream and returns the new position within the current stream.

Declaration

```
// C#
public long Seek(long offset, SeekOrigin origin);
```

Parameters

- *offset*

  A byte offset relative to origin.

  - If *offset* is negative, the new position precedes the position specified by *origin* by the number of bytes specified by *offset*.
  - If offset is zero, the new position is the position specified by *origin*.
  - If *offset* is positive, the new position follows the position specified by *origin* by the number of bytes specified by *offset*.

- *origin*

  A value of type `SeekOrigin` indicating the reference point used to obtain the new position.

Return Value

The new `Position` within the current stream.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object

Remarks

Use the `CanSeek` property to determine whether the current instance supports seeking. Seeking to any location beyond the length of the stream is supported.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlStream Class
- OracleXmlStream Members

OracleXmlType Class

An `OracleXmlType` object represents an Oracle `XMLType` instance.

Class Inheritance

```
Object
  OracleXmlType
```

Declaration

```
// C#
public sealed class OracleXmlType : IDisposable, ICloneable
```

Thread Safety

All public static methods are thread-safe, although instance methods do not guarantee thread safety.

Remarks

`OracleXmlType` objects can be used for well-formed XML documents with or without XML schemas or XML fragments.

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

This class can only be used with Oracle9*i* Release 2 (9.2) or higher.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Members
- OracleXmlType Constructors
- OracleXmlType Static Methods
- OracleXmlType Instance Properties
- OracleXmlType Instance Methods

OracleXmlType Members

`OracleXmlType` members are listed in the following tables:

OracleXmlType Constructors

The `OracleXmlType` constructors are listed in Table 5–187.

Table 5–187 OracleXmlType Constructors

Constructor	Description
OracleXmlType Constructors	Creates an instance of the `OracleXmlType` class (Overloaded)

OracleXmlType Static Methods

The `OracleXmlType` static methods are listed in Table 5–188.

Table 5–188 OracleXmlType Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

OracleXmlType Instance Properties

The `OracleXmlType` instance properties are listed in Table 5–189.

Table 5–189 OracleXmlType Instance Properties

Properties	Description
Connection	Indicates the `OracleConnection` that is used to retrieve and store XML data in the `OracleXmlType`
IsEmpty	Indicates whether or not the `OracleXmlType` is empty
IsFragment	Indicates whether the XML data is a collection of XML elements or a well-formed XML document
IsSchemaBased	Indicates whether or not the XML data represented by the `OracleXmlType` is based on an XML schema
RootElement	Represents the name of the top-level element of the schema-based XML data contained in the `OracleXmlType`
Schema	Represents the XML schema of the XML data contained in the `OracleXmlType`

Table 5–189 OracleXmlType Instance Properties (Cont.)

Properties	Description
SchemaUrl	Represents **URL** in the database for the XML schema of the XML data contained in the `OracleXmlType`.

OracleXmlType Instance Methods

The `OracleXmlType` instance methods are listed in Table 5–190.

Table 5–190 OracleXmlType Instance Methods

Methods	Description
Clone	Creates a copy of the `OracleXmlType` instance
Dispose	Releases the resources allocated by this `OracleXmlType` object
`Equals`	Inherited from `Object`
Extract	Extracts a subset from the XML data using the given XPath expression (Overloaded)
`GetHashCode`	Inherited from `Object`
GetStream	Returns an instance of `OracleXmlStream` which provides a read-only stream of the XML data stored in this `OracleXmlType` instance
`GetType`	Inherited from `Object`
GetXmlDocument	Returns a `XmlDocument` object containing the XML data stored in this `OracleXmlType` instance
GetXmlReader	Returns a `XmlTextReader` object that can be used to manipulate XML data directly using the .NET Framework classes and methods
IsExists	Checks for the existence of a particular set of nodes identified by the given XPath expression in the XMLdata (Overloaded)
`ToString`	Inherited from `Object`
Transform	Transforms the `OracleXmlType` into another `OracleXmlType` instance using the given XSL document (Overloaded)
Update	Updates the XML node or fragment identified by the given XPath expression in the current `OracleXmlType` instance (Overloaded)

Table 5–190 OracleXmlType Instance Methods (Cont.)

Methods	Description
Validate	Validates whether the XML data in the `OracleXmlType` object conforms to the given XML schema.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class

OracleXmlType Constructors

`OracleXmlType` constructors create instances of the `OracleXmlType` class.

Overload List:

- OracleXmlType(OracleClob)

  This constructor creates an instance of the `OracleXmlType` class using the XML data contained in an `OracleClob` object.

- OracleXmlType(OracleConnection, string)

  This constructor creates an instance of the `OracleXmlType` class using the XML data contained in the .NET `String`.

- OracleXmlType(OracleConnection, XmlReader)

  This constructor creates an instance of the `OracleXmlType` class using the contents of the .NET `XmlReader` object.

- OracleXmlType(OracleConnection, XmlDocument)

  This constructor creates an instance of the `OracleXmlType` object using the contents of the XML **DOM** document in the .NET `XmlDocument` object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

OracleXmlType(OracleClob)

This constructor creates an instance of the `OracleXmlType` class using the XML data contained in an `OracleClob` object.

Declaration

```
// C#
public OracleXmlType(OracleClob oraClob);
```

Parameters

- *`oraClob`*

  An `OracleClob` object.

Exceptions

`ArgumentNullException` - The `OracleClob` object is null.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The `CLOB` data depends on a valid connection object and the new `OracleXMLType` uses the `OracleConnection` in the `OracleClob` object to store data for the current instance.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

OracleXmlType(OracleConnection, string)

This constructor creates an instance of the `OracleXmlType` class using the XML data contained in the .NET `String`.

Declaration

```
// C#
public OracleXmlType(OracleConnection con, string xmlData);
```

Parameters

- *con*

  An OracleConnection object.

- *xmlData*

  A string containing the XML data.

Exceptions

ArgumentNullException - The OracleConnection object is null.

ArgumentException - The xmlData argument is an empty string.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The new OracleXmlType uses the given OracleConnection object to store data for the current instance.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

OracleXmlType(OracleConnection, XmlReader)

This constructor creates an instance of the OracleXmlType class using the contents of the .NET XmlReader object.

Declaration

```
// C#
public OracleXmlType(OracleConnection con, XmlReader reader);
```

Parameters

- *con*

  An OracleConnection object.

- *reader*

  An XmlReader object.

Exceptions

`ArgumentNullException` - The `OracleConnection` object is null.

`ArgumentException` - The *`reader`* argument contains no data.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The new `OracleXMLType` uses the given `OracleConnection` object to store data for the current instance.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

OracleXmlType(OracleConnection, XmlDocument)

This constructor creates an instance of the `OracleXmlType` object using the contents of the XML **DOM** document in the .NET `XmlDocument` object.

Declaration

```
// C#
public OracleXmlType(OracleConnection con, XmlDocument domDoc);
```

Parameters

- *`con`*

  An `OracleConnection` object.

- *`domDoc`*

  An XML document.

Exceptions

`ArgumentNullException` - The `OracleConnection` object is null.

`ArgumentException` - The *`domDoc`* argument contains no data.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The new `OracleXMLType` uses the given `OracleConnection` object to store data for the current instance.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

OracleXmlType Static Methods

The `OracleXmlType` static methods are listed in Table 5–191.

Table 5–191 OracleXmlType Static Methods

Methods	Description
`Equals`	Inherited from `Object` (Overloaded)

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

OracleXmlType Instance Properties

The `OracleXmlType` instance properties are listed in Table 5–192.

Table 5–192 OracleXmlType Instance Properties

Properties	Description
Connection	Indicates the `OracleConnection` that is used to retrieve and store XML data in the `OracleXmlType`
IsEmpty	Indicates whether or not the `OracleXmlType` is empty
IsFragment	Indicates whether the XML data is a collection of XML elements or a well-formed XML document
IsSchemaBased	Indicates whether or not the XML data represented by the `OracleXmlType` is based on an XML schema

Table 5–192 OracleXmlType Instance Properties (Cont.)

Properties	Description
RootElement	Represents the name of the top-level element of the schema-based XML data contained in the `OracleXmlType`
Schema	Represents the XML schema of the XML data contained in the `OracleXmlType`
SchemaUrl	Represents **URL** in the database for the XML schema of the XML data contained in the `OracleXmlType`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Connection

This property indicates the `OracleConnection` that is used to retrieve and store XML data in the `OracleXmlType`.

Declaration

```
// C#
public OracleConnection Connection {get;}
```

Property Value

An `OracleConnection` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

The connection must explicitly be opened by the user before creating or using `OracleXmlType`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

IsEmpty

This property indicates whether or not the `OracleXmlType` is empty.

Declaration

```
// C#
public bool IsEmpty {get;}
```

Property Value

Returns `true` if the `OracleXmlType` represents an empty XML document. Returns `false` otherwise.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

IsFragment

This property indicates whether the XML data is a collection of XML elements or a well-formed XML document.

Declaration

```
// C#
public bool IsFragment {get;}
```

Property Value

Returns `true` if the XML data contained in the `OracleXmlType` object is a collection of XML elements with no root element. Returns `false` otherwise.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

IsSchemaBased

This property indicates whether or not the XML data represented by the `OracleXmlType` is based on an XML schema.

Declaration

```
// C#
public bool IsSchemaBased {get;}
```

Property Value

Returns `true` if the XML data represented by the `OracleXmlType` is based on an XML schema. Returns `false` otherwise.

Exceptions

`ObjectDisposedException` - The object is already disposed.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

RootElement

This property represents the name of the top-level or root element of the schema-based XML data contained in the `OracleXmlType`.

Declaration

```
// C#
public string RootElement{get;}
```

Property Value

A string that represents the name of the top-level or root element of the XML data contained in the `OracleXmlType`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

If the `OracleXmlType` instance contains non-schema based XML data, this property returns an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Schema

This property represents the XML schema for the XML data contained in the `OracleXmlType`.

Declaration

```
// C#
public OracleXmlType Schema {get;}
```

Property Value

An `OracleXmlType` instance that represents the XML schema for the XML data contained in the `OracleXmlType`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

If the `OracleXmlType` instance contains non-schema based XML data, this property returns an `OracleXmlType` instance representing an empty XML document.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

SchemaUrl

This property represents the **URL** in the database for the XML schema of the XML data contained in the `OracleXmlType`.

Declaration

```
// C#
public string SchemaUrl {get;}
```

Property Value

A string that represents the URL in the database for the XML schema of the XML data.

Exceptions

`ObjectDisposedException` - The object is already disposed.

Remarks

If the `OracleXmlType` instance contains non-schema based XML data, this property returns an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Value

This property returns the XML data starting from the first character in the current instance as a `string`.

Declaration

```
// C#
public string RootElement{get;}
```

Property Value

The entire XML data as a `string`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

OracleXmlType Instance Methods

The `OracleXmlType` instance methods are listed in Table 5–193.

Table 5–193 OracleXmlType Instance Methods

Methods	Description
Clone	Creates a copy of the `OracleXmlType` instance
Dispose	Releases the resources allocated by this `OracleXmlType` object
`Equals`	Inherited from `Object`
Extract	Extracts a subset from the XML data using the given XPath expression (Overloaded)
`GetHashCode`	Inherited from `Object`
GetStream	Returns an instance of `OracleXmlStream` which provides a read-only stream of the XML data stored in this `OracleXmlType` instance
`GetType`	Inherited from `Object`
GetXmlDocument	Returns a `XmlDocument` object containing the XML data stored in this `OracleXmlType` instance

Table 5–193 OracleXmlType Instance Methods (Cont.)

Methods	Description
GetXmlReader	Returns a `XmlTextReader` object that can be used to manipulate XML data directly using the .NET Framework classes and methods
IsExists	Checks for the existence of a particular set of nodes identified by the given XPath expression in the XMLdata (Overloaded)
`ToString`	Inherited from `Object`
Transform	Transforms the `OracleXmlType` into another `OracleXmlType` instance using the given XSL document (Overloaded)
Update	Updates the XML node or fragment identified by the given XPath expression in the current `OracleXmlType` instance (Overloaded)
Validate	Validates whether the XML data in the `OracleXmlType` object conforms to the given XML schema.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Clone

This method creates a copy of this `OracleXmlType` instance.

Declaration

```
// C#
public object Clone();
```

Implements

`ICloneable`

Return Value

An `OracleXmlType` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Dispose

This method releases the resources allocated by this object.

Declaration

```
// C#
public void Dispose();
```

Implements

`IDisposable`

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Extract

This method extracts a subset from the XML data using the given XPath expression.

Overload List:

- Extract(string, string)

  This method extracts a subset from the XML data represented by the `OracleXmlType` object using the given XPath expression and a string parameter for namespace resolution.

- Extract(string, XmlNameSpaceManager)

This method extracts a subset from the XML data represented by the `OracleXmlType` object, using the given XPath expression and a .NET `XmlNameSpaceManager` object for namespace resolution.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Extract(string, string)

This method extracts a subset from the XML data represented by the `OracleXmlType` object using the given XPath expression and a string parameter for namespace resolution.

Declaration

```
// C#
public OracleXmlType Extract(string xpathExpr, string nsMap);
```

Parameters

- *xpathExpr*

  The XPath expression.

- *nsMap*

  The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null. For example:

  ```
  xmlns:nsi"=http://www.company1.com" xmlns:nsz="http://www.company2.com"
  ```

Return Value

An `OracleXmlType` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xpathExpr* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Extract(string, XmlNameSpaceManager)

This public method extracts a subset from the XML data represented by the `OracleXmlType` object, using the given XPath expression and a .NET `XmlNameSpaceManager` object for namespace resolution.

Declaration

```
// C#
public OracleXmlType Extract(string xpathExpr, XmlNameSpaceManager nsMgr);
```

Parameters

- *xpathExpr*

  The XPath expression.

- *nsMgr*

  The .NET `XmlNameSpaceManager` object used for namespace resolution of the XPath expression. *nsMgr* can be null.

Return Value

An `OracleXmlType`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xpathExpr* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

GetStream

This public method returns an instance of `OracleXmlStream` which provides a read-only stream of the XML data stored in this `OracleXmlType` instance.

Declaration

```
// C#
public Stream GetStream();
```

Return Value

A `Stream` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

GetXmlDocument

This public method returns a `XmlDocument` object containing the XML data stored in this `OracleXmlType` instance.

Declaration

```
// C#
public XmlDocument GetXmlDocument();
```

Return Value

An `XmlDocument` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The XML data in the `XmlDocument` object is a copy of the XML data in the `OracleXmlType` instance and modifying it does not automatically modify the XML data in the `OracleXmlType` instance. The `XmlDocument` instance returned has the `PreserveWhitespace` property set to `true`.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

GetXmlReader

This public method returns a `XmlTextReader` object that can be used to manipulate XML data directly using the .NET Framework classes and methods.

Declaration

```
// C#
public XmlTextReader GetXmlReader();
```

Return Value

An `XmlTextReader` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The `XmlTextReader` is a read-only, forward-only representation of the XML data stored in the `OracleXmlType` instance.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

IsExists

`IsExists` checks for the existence of a particular set of nodes identified by the XPath expression in the XML data.

Overload List:

- IsExists(string, string)

  This method checks for the existence of a particular set of nodes identified by the XPath expression in the XML data represented by the current `OracleXmlType` instance using a string parameter for namespace resolution.

- IsExists(string, XmlNameSpaceManager)

  This method checks for the existence of a particular set of nodes identified by the XPath expression in the XML document represented by the current `OracleXmlType` instance using a .NET `XmlNameSpaceManager` object for namespace resolution.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

IsExists(string, string)

This method checks for the existence of a particular set of nodes identified by the XPath expression in the XML data represented by the current `OracleXmlType` instance using a string parameter for namespace resolution.

Declaration

```
// C#
public bool IsExists(string xpathExpr, string nsMap);
```

Parameters

- *xpathExpr*

  The XPath expression.

- *nsMap*

  The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null.

Return Value

Returns `true` if the required set of nodes exists; otherwise, returns `false`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xpathExpr* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

IsExists(string, XmlNameSpaceManager)

This method checks the existence of a particular set of nodes identified by the XPath expression in the XML document represented by the current `OracleXmlType` instance using a .NET `XmlNameSpaceManager` object for namespace resolution.

Declaration

```
// C#
public bool IsExists(string xpathExpr, XmlNameSpaceManager nsMgr);
```

Parameters

- *xpathExpr*

The XPath expression.

- *nsMgr*

  The .NET `XmlNameSpaceManager` object used for namespace resolution of the XPath expression. *nsMgr* can be null.

Return Value

Returns `true` if the required set of nodes exists; otherwise, returns `false`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The `xpathExpr` is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Transform

This method transforms the `OracleXmlType` into another `OracleXmlType` instance using the given XSL document.

Overload List:

- Transform(OracleXmlType, string)

  This method transforms the current `OracleXmlType` instance into another `OracleXmlType` instance using the given XSL document (as an `OracleXmlType` object) and a string of XSLT parameters.

- Transform(string, string)

This public method transforms the current `OracleXmlType` instance into another `OracleXmlType` instance using the given XSL document and a string of XSLT parameters.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Transform(OracleXmlType, string)

This method transforms the current `OracleXmlType` instance into another `OracleXmlType` instance using the given XSL document and a string of XSLT parameters.

Declaration

```
// C#
public OracleXmlType Transform(OracleXmlType xsldoc, string paramMap);
```

Parameters

- *xsldoc*

  The XSL document as an `OracleXmlType` object.

- *paramMap*

  A string which provides the parameters for the XSL document.

  For this release, *paramMap* is ignored.

Return Value

An `OracleXmlType` object containing the transformed XML document.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xsldoc* parameter is null.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Transform(string, string)

This method transforms the current `OracleXmlType` instance into another `OracleXmlType` instance using the given XSL document and a string of XSLT parameters.

Declaration

```
// C#
public OracleXmlType Transform(string xsldoc, string paramMap);
```

Parameters

- *`xsldoc`*

  The XSL document to be used for XSLT.

- *`paramMap`*

  A string which provides the parameters for the XSL document.

  For this release, `paramMap` is ignored.

Return Value

An `OracleXmlType` object containing the transformed XML document.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *`xsldoc`* parameter is null.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Update

This method updates the XML node or fragment identified by the given XPath expression in the current `OracleXmlType` instance.

Overload List:

- Update(string, string, string)

  This method updates the XML nodes identified by the given XPath expression with the given string value and a string parameter for namespace resolution.

- Update(string, XmlNameSpaceManager, string)

  This method updates the XML nodes identified by the given XPath expression with the given string value and a .NET `XmlNameSpaceManager` object for namespace resolution.

- Update(string, string, OracleXmlType)

  This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given `OracleXmlType` value and a string parameter for namespace resolution.

- Update(string, XmlNameSpaceManager, OracleXmlType)

  This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given `OracleXmlType` value and a .NET `XmlNameSpaceManager` object for namespace resolution.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Update(string, string, string)

This method updates the XML nodes identified by the given XPath expression with the given string value and a string parameter for namespace resolution.

Declaration

```
// C#
public void Update(string xpathExpr, string nsMap, string value);
```

Parameters

- *xpathExpr*

  The XPath expression that identifies the nodes to update.

- *nsMap*

  The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null. For example:

  ```
  xmlns:nsi"=http://www.company1.com" xmlns:nsz="http://www.company2.com"
  ```

- *value*

  The new value as a `string`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xpathExpr* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Update(string, XmlNameSpaceManager, string)

This method updates the XML nodes identified by the given XPath expression with the given string value and a .NET `XmlNameSpaceManager` object for namespace resolution.

Declaration

```
// C#
public void Update(string xpathExpr, XmlNameSpaceManager nsMgr, string value);
```

Parameters

- *xpathExpr*

  The XPath expression that identifies the nodes to update.

- *nsMgr*

  The .NET `XmlNameSpaceManager` object used for namespace resolution of the XPath expression. *nsMgr* can be null.

- *value*

  The new value as a `string`.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xpathExpr* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Update(string, string, OracleXmlType)

This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given `OracleXmlType` value and a string parameter for namespace resolution.

Declaration

```
// C#
public void Update(string xpathExpr, string nsMap, OracleXmlType value);
```

Parameters

- *xpathExpr*

The XPath expression that identifies the nodes to update.

- *nsMap*

  The string parameter used for namespace resolution of the XPath expression. *nsMap* has zero or more namespaces separated by spaces. *nsMap* can be null.

- *value*

  The new value as an `OracleXmlType` object.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`ArgumentNullException` - The *xpathExpr* is null or zero-length.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Update(string, XmlNameSpaceManager, OracleXmlType)

This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given `OracleXmlType` value and a .NET `XmlNameSpaceManager` object for namespace resolution.

Declaration

```
// C#
public void Update(string xpathExpr, XmlNameSpaceManager nsMgr, OracleXmlType
value);
```

Parameters

- *xpathExpr*

  The XPath expression that identifies the nodes to update.

- *nsMgr*

  The .NET XmlNameSpaceManager object used for namespace resolution of the XPath expression. *nsMgr* can be null.

- *value*

  The new value as an OracleXmlType object.

Exceptions

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The *xpathExpr* is null or zero-length.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The default namespace is ignored if its value is an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Validate

This methods validates whether the XML data in the OracleXmlType object conforms to the given XML schema.

Declaration

```
// C#
public bool Validate(String schemaUrl);
```

Parameters

- schemaUrl

  A string representing the **URL** in the database of the XML schema.

Return Value

Returns true if the XML data conforms to the XML schema; otherwise, returns false.

Exceptions

`ObjectDisposedException` - The object is already disposed.

`InvalidOperationException` - The `OracleConnection` is not open or has been closed during the lifetime of the object.

`ArgumentNullException` - The *`schemaUrl`* argument is null or an empty string.

See Also:

- Oracle.DataAccess.Types Namespace (ODP.NET Types)
- OracleXmlType Class
- OracleXmlType Members

Glossary

assembly

Assembly is Microsoft's term for the module that is created when a DLL or .EXE is complied by a .NET compiler.

BFILES

External binary files that exist outside the database tablespaces residing in the operating system. BFILES are referenced from the database semantics, and are also known as external LOBs.

Binary Large Object (BLOB)

A large object datatype whose content consists of binary data. Additionally, this data is considered raw as its structure is not recognized by the database.

Character Large Object (CLOB)

The LOB datatype whose value is composed of character data corresponding to the database character set. A `CLOB` may be indexed and searched by the Oracle Text search engine.

data provider

As the term is used with Oracle Data Provider for .NET, a data provider is the connected component in the ADO.NET model and transfers data between a data source and the `DataSet`.

dirty writes

Dirty writes means writing uncommitted or dirty data.

DDL

DDL refers to data definition language, which includes statements defining or changing data structure.

DOM

Document Object Model (DOM) is an application program interface (API) for HTML and XML documents. It defines the logical structure of documents and the way that a document is accessed and manipulated.

flush

Flush or flushing refers to recording changes (that is, sending modified data) to the database.

instantiate

A term used in object-based languages such as C# to refer to the creation of an object of a specific class.

Large Object (LOB)

The class of SQL datatype that is further divided into internal LOBs and external LOBs. Internal LOBs include `BLOB`s, `CLOB`s, and `NCLOB`s while external LOBs include `BFILE`s.

Microsoft .NET Framework Class Library

The Microsoft .NET Framework Class Library provides the classes for the .NET framework model.

namespace

- .NET:

  A namespace is naming device for grouping related types. More than one namespace can be contained in an assembly.

- XML Documents:

  A namespace describes a set of related element names or attributes within an XML document.

National Character Large Object (NCLOB)

The LOB datatype whose value is composed of character data corresponding to the database national character set.

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.

OracleDataReader

An `OracleDataReader` is a read-only, forward-only result set.

Oracle XML Database (XML DB)

Oracle XML DB is the name for a distinct group of technologies related to high-performance XML storage and retrieval that are available within the Oracle database. Oracle XML DB is not a separate server.

Oracle XML DB is based on the W3C XML data model.

PL/SQL

Oracle's procedural language extension to SQL.

primary key

The column or set of columns included in the definition of a table's PRIMARY KEY constraint.

reference semantics

Reference semantics indicates that assignment is to a reference (an address such as a pointer) rather than to a value. See **value semantics**.

result set

The output of a SQL query, consisting of one or more rows of data.

Safe Type Mapping

Safe Type Mapping allows the `OracleDataAdapter` to populate a `DataSet` with .NET type representations of Oracle data without any data or precision loss.

savepoint

A point in the workspace to which operations can be rolled back.

stored procedure

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

Transparent Application Failover (TAF)

Transparent Application Failover is a runtime failover for high-availability environments. It enables client applications to automatically reconnect to the database if the connection fails. This reconnect happens automatically from within the Oracle Call Interface (OCI) library.

Unicode

Unicode is a universal encoded character set that enables information from any language to be stored using a single character set.

URL

URL (Universal Resource Locator).

value semantics

Value semantics indicates that assignment copies the value, not the reference or address (such as a pointer). See **reference semantics**.

XPath

XML Path Language (XPath), based on a W3C recommendation, is a language for addressing parts of an XML document. It is designed to be used by both XSLT and XPointer. It can be used as a searching or query language as well as in hypertext linking.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

X