## Contents

Send Us Your Comments .......................................................................................................................... xiii

Preface ........................................................................................................................................................... xv
  Audience .................................................................................................................................................... xv
  Documentation Accessibility ..................................................................................................................... xv
  Structure ................................................................................................................................................... xvi
  Related Documents ................................................................................................................................. xvii
  Conventions ............................................................................................................................................. xviii

What’s New in Oracle Data Provider for .NET? ......................................................................................... xxiii
  New Features in Oracle Data Provider for .NET Release 10.1.0.3 ....................................................... xxiii
  New Features in Oracle Data Provider for .NET Release 10.1 ............................................................... xxiv
  New Features in Oracle Data Provider for .NET Release 9.2.0.4 ....................................................... xxiv

Volume 1

1 **Introducing Oracle Data Provider for .NET**
   Overview of Oracle Data Provider for .NET (ODP.NET) ................................................................. 1-1
   ODP.NET Assembly ............................................................................................................................... 1-1
   Oracle.DataAccess.Client Namespace .................................................................................................. 1-1
   Oracle.DataAccess.Client Classes ................................................................................................. 1-1
   Oracle.DataAccess.Client Enumerations ........................................................................................... 1-3
   Oracle.DataAccess.Types Namespace ............................................................................................... 1-4
   Oracle.DataAccess.Types Structures .............................................................................................. 1-4
   Oracle.DataAccess.Types Exceptions ............................................................................................... 1-4
   Oracle.DataAccess.Types Classes ....................................................................................................... 1-4
   Using ODP.NET in a Simple Application ......................................................................................... 1-5

2 **Installing and Configuring Oracle Data Provider for .NET**
   System Requirements ......................................................................................................................... 2-1
   Installing Oracle Data Provider for .NET .......................................................................................... 2-2
   File Locations .................................................................................................................................... 2-2
3 Features of Oracle Data Provider for .NET

Connecting to the Oracle Database ................................................................. 3-1
  Connection String Attributes ................................................................. 3-1
  Connection Pooling ........................................................................... 3-3
    Using Connection Pooling ......................................................... 3-4
  Operating System Authentication .................................................... 3-5
  Privileged Connections .................................................................... 3-5
  Password Expiration ......................................................................... 3-6
  Proxy Authentication .......................................................................... 3-7
  Transparent Application Failover (TAF) Callback Support ............... 3-8
    TAF Notification ........................................................................ 3-8
    When Failover Occurs .................................................................. 3-8
    Registering an Event Handler for Failover ................................... 3-9

OracleCommand Object .............................................................................. 3-10
  Transaction ....................................................................................... 3-10
  Parameter Binding ............................................................................. 3-10
    Datatypes BINARY_FLOAT and BINARY_DOUBLE ...................... 3-11
    OracleDbType Enumeration Type .................................................. 3-11
    Inference of DbType, OracleDbType, and .NET Types .............. 3-12
    PL/SQL Associative Array ............................................................. 3-15
    Array Binding .................................................................................. 3-18
  Statement Caching ............................................................................... 3-21
    Statement Caching Connection String Attributes ....................... 3-21
    Statement Caching Methods and Properties ................................ 3-21
    Connections and Statement Caching ........................................... 3-22
    Pooling and Statement Caching .................................................... 3-22

ODP.NET Types Overview ........................................................................... 3-22

Obtaining Data From an OracleDataReader ............................................... 3-23
  Typed OracleDataReader Accessors ............................................... 3-23
    .NET Type Accessors .................................................................. 3-23
    ODP.NET Type Accessors .............................................................. 3-26
  Obtaining LONG and LONG RAW Data ......................................... 3-27
  Obtaining LOB Data .......................................................................... 3-27
    Methods Supported or Not Supported for InitialLOBFetchSize .... 3-28
    LOB Data Fetching Considerations ............................................ 3-28
    Performance ................................................................................... 3-29
  Controlling the Number of Rows Fetched in One Server Round-Trip ... 3-29
    Use of FetchSize ........................................................................ 3-29
    Fine-Tuning FetchSize ................................................................ 3-29
    Using the RowSize Property ....................................................... 3-30

PL/SQL REF CURSOR and OracleRefCursor ............................................. 3-30
  Obtaining an OracleRefCursor ......................................................... 3-30
  Obtaining a REF CURSOR ............................................................... 3-31
  Populating an OracleDataReader from a REF CURSOR .................... 3-31
  Populating the DataSet From a REF CURSOR .................................. 3-31
  Populating an OracleRefCursor From a REF CURSOR ..................... 3-32
Updating a DataSet Obtained From a REF CURSOR ................................................................. 3-32
Behavior of ExecuteScalar Method for REF CURSOR .......................................................... 3-32

**LOB Support** .......................................................................................................................... 3-32
Large Character and Large Binary Datatypes ......................................................................... 3-33
Oracle Data Provider for .NET LOB Objects ............................................................................. 3-33
Updating LOBs Using a DataSet ............................................................................................... 3-34
Updating LOBs Using OracleCommand and OracleParameter ............................................ 3-34
Updating LOBs Using ODP.NET LOB Objects ........................................................................ 3-34
Temporary LOBs .......................................................................................................................... 3-35

**ODP.NET XML Support** .......................................................................................................... 3-35
Supported XML Features ........................................................................................................... 3-36
OracleXmlType and Connection Dependency ........................................................................ 3-37
Updating XMLType Data in the Database Server ...................................................................... 3-37
  Updating with DataSet, OracleDataAdapter, and OracleCommandBuilder ...................... 3-37
  Updating with OracleCommand and OracleParameter ...................................................... 3-38
Updating XML Data in OracleXmlType .................................................................................... 3-39
Special Characters in XML ......................................................................................................... 3-39
Retrieving Query Result Set as XML ....................................................................................... 3-39
  Handling Date and Time Format ......................................................................................... 3-39
  Special Characters in Column Data .................................................................................... 3-40
  Special Characters In Table or View Name .......................................................................... 3-41
  Case-Sensitivity in Column Name to XML Element Name Mapping .............................. 3-41
  Column Name to XML Element Name Mapping .................................................................. 3-42
  Object-Relational Data .......................................................................................................... 3-43
  NULL values ............................................................................................................................. 3-44
Data Manipulation Using XML ................................................................................................ 3-44
  Handling of Date and Time Format .................................................................................... 3-44
  Saving Changes Using XML ............................................................................................... 3-44
  Special Characters in Column Data .................................................................................... 3-45
  Special Characters in Table or View Name ........................................................................ 3-45
  Case-Sensitivity in XML Element Name to Column Name Mapping ............................ 3-46
  XML Element Name to Column Name Mapping ............................................................... 3-46
  Object-Relational Data .......................................................................................................... 3-48
  Multiple Tables ...................................................................................................................... 3-48
  Commits .................................................................................................................................. 3-48

**OracleDataAdapter Safe Type Mapping** ............................................................................. 3-49
Potential Data Loss .................................................................................................................... 3-49
SafeMapping Property .............................................................................................................. 3-50
Using Safe Type Mapping ......................................................................................................... 3-50

**OracleDataAdapter Requery Property** ............................................................................... 3-51

**Guaranteeing Uniqueness in Updating DataSet to Database** ................................................ 3-51
What Constitutes Uniqueness in DataRows? ............................................................................ 3-52
Configuring PrimaryKey and Constraints Properties ............................................................. 3-52
Updating Without PrimaryKey and Constraints Configuration ............................................. 3-53
Globalization Support ................................................................................................................. 3-53
  Globalization Settings .................................................................................................................. 3-53
    Client Globalization Settings ................................................................................................. 3-54
    Session Globalization Settings .............................................................................................. 3-54
    Thread-Based Globalization Settings ....................................................................................... 3-55
  Globalization-Sensitive Operations ............................................................................................ 3-56
    Operations Dependent on Client Computer's Globalization Settings ..................................... 3-56
    Operations Dependent on Thread Globalization Settings ....................................................... 3-56
    Operations Sensitive to Session Globalization Parameters .................................................... 3-57
Debug Tracing ............................................................................................................................... 3-57
  Registry Settings for Tracing Calls ............................................................................................. 3-57
    TraceFileName .......................................................................................................................... 3-57
    TraceLevel ............................................................................................................................... 3-57
    TraceOption ............................................................................................................................. 3-58

4 Oracle Data Provider for .NET Classes

OracleCommand Class .................................................................................................................. 4-2
  OracleCommand Members ......................................................................................................... 4-4
  OracleCommand Constructors ................................................................................................... 4-7
  OracleCommand Static Methods ............................................................................................... 4-9
  OracleCommand Properties ..................................................................................................... 4-10
  OracleCommand Public Methods .............................................................................................. 4-24
OracleCommandBuilder Class .................................................................................................... 4-38
  OracleCommandBuilder Members ............................................................................................. 4-40
  OracleCommandBuilder Constructors ....................................................................................... 4-42
  OracleCommandBuilder Static Methods .................................................................................... 4-44
  OracleCommandBuilder Properties ........................................................................................... 4-47
  OracleCommandBuilder Public Methods .................................................................................... 4-49
  OracleCommandBuilder Events .................................................................................................. 4-52
  OracleCommandBuilder Event Delegates ................................................................................... 4-53
OracleConnection Class .............................................................................................................. 4-54
  OracleConnection Members ...................................................................................................... 4-56
  OracleConnection Constructors ................................................................................................. 4-59
  OracleConnection Static Methods ............................................................................................. 4-61
  OracleConnection Properties .................................................................................................... 4-62
  OracleConnection Public Methods ............................................................................................. 4-71
  OracleConnection Events .......................................................................................................... 4-83
  OracleConnection Event Delegates ............................................................................................. 4-85
OracleDataAdapter Class ............................................................................................................. 4-86
  OracleDataAdapter Members ..................................................................................................... 4-88
  OracleDataAdapter Constructors ............................................................................................... 4-90
  OracleDataAdapter Static Methods ............................................................................................ 4-93
  OracleDataAdapter Properties ................................................................................................... 4-94
  OracleDataAdapter Public Methods ............................................................................................ 4-99
  OracleDataAdapter Events ......................................................................................................... 4-104
  OracleDataAdapter Event Delegates ......................................................................................... 4-108
5 Oracle Data Provider for .NET XML-Related Classes

OracleXmlCommandType Enumeration ................................................................. 5-2
OracleXmlQueryProperties Class ....................................................................... 5-3
  OracleXmlQueryProperties Members ............................................................... 5-6
  OracleXmlQueryProperties Constructor ......................................................... 5-7
  OracleXmlQueryProperties Properties ........................................................... 5-8
  OracleXmlQueryProperties Public Methods .................................................... 5-11
OracleXmlSaveProperties Class ....................................................................... 5-12
  OracleXmlSaveProperties Members ............................................................... 5-15
  OracleXmlSaveProperties Constructor ......................................................... 5-16
  OracleXmlSaveProperties Properties ............................................................. 5-17
  OracleXmlSaveProperties Public Methods .................................................... 5-21
OracleXmlStream Class ..................................................................................... 5-22
  OracleXmlStream Members ............................................................................ 5-23
  OracleXmlStream Constructor ....................................................................... 5-25
  OracleXmlStream Static Methods .................................................................. 5-26
  OracleXmlStream Instance Properties ........................................................... 5-27
  OracleXmlStream Instance Methods ............................................................... 5-31
OracleXmlType Class ......................................................................................... 5-36
  OracleXmlType Members ............................................................................... 5-37
  OracleXmlType Constructors ........................................................................ 5-39
  OracleXmlType Static Methods ..................................................................... 5-42
  OracleXmlType Instance Properties ............................................................... 5-43
  OracleXmlType Instance Methods ................................................................. 5-48

6 Oracle Data Provider for .NET Globalization Classes

OracleGlobalization Class .................................................................................. 6-2
  OracleGlobalization Members ...................................................................... 6-4
  OracleGlobalization Static Methods ............................................................. 6-6
  OracleGlobalization Properties .................................................................... 6-12
  OracleGlobalization Public Methods ............................................................. 6-22
7 Oracle Data Provider for .NET Failover Classes

OracleFailoverEventArgs Class ................................................................. 7-2
OracleFailoverEventArgs Members ....................................................... 7-4
OracleFailoverEventArgs Static Methods ............................................ 7-5
OracleFailoverEventArgs Properties .................................................. 7-6
OracleFailoverEventArgs Public Methods ............................................ 7-7

OracleFailoverEventHandler Delegate .................................................. 7-8
FailoverEvent Enumeration .................................................................... 7-10
FailoverReturnCode Enumeration ....................................................... 7-11
FailoverType Enumeration .................................................................... 7-12

Volume 2

8 Oracle Data Provider for .NET Types Classes

OracleBFile Class ................................................................................... 8-2
OracleBFile Members ........................................................................... 8-4
OracleBFile Constructors ..................................................................... 8-7
OracleBFile Static Fields ....................................................................... 8-9
OracleBFile Static Methods ................................................................. 8-10
OracleBFile Instance Properties ......................................................... 8-11
OracleBFile Instance Methods ............................................................ 8-17

OracleBlob Class .................................................................................. 8-35
OracleBlob Members ............................................................................ 8-37
OracleBlob Constructors ..................................................................... 8-40
OracleBlob Static Fields ....................................................................... 8-42
OracleBlob Static Methods ................................................................. 8-43
OracleBlob Instance Properties ......................................................... 8-44
OracleBlob Instance Methods ............................................................. 8-49

OracleClob Class .................................................................................. 8-69
OracleClob Members ............................................................................ 8-71
OracleClob Constructors ..................................................................... 8-74
OracleClob Static Fields ....................................................................... 8-76
OracleClob Static Methods ................................................................. 8-77
OracleClob Instance Properties ......................................................... 8-78
OracleClob Instance Methods ............................................................. 8-84

OracleRefCursor Class .......................................................................... 8-109
OracleRefCursor Members ................................................................... 8-111
OracleRefCursor Static Methods ......................................................... 8-112
OracleRefCursor Properties ................................................................. 8-113
OracleRefCursor Instance Methods .................................................... 8-114
9 Oracle Data Provider for .NET Types Structures

OracleBinary Structure ................................................................. 9-2
  OracleBinary Members .............................................................. 9-4
  OracleBinary Constructor ........................................................ 9-7
  OracleBinary Static Fields ....................................................... 9-8
  OracleBinary Static Methods ................................................... 9-9
  OracleBinary Static Operators ................................................ 9-15
  OracleBinary Static Type Conversion Operators ...................... 9-21
  OracleBinary Properties ......................................................... 9-23
  OracleBinary Instance Methods .............................................. 9-26

OracleDate Structure ................................................................. 9-29
  OracleDate Members ............................................................... 9-31
  OracleDate Constructors ....................................................... 9-34
  OracleDate Static Fields ........................................................ 9-39
  OracleDate Static Methods ..................................................... 9-41
  OracleDate Static Operators ................................................... 9-47
  OracleDate Static Type Conversions ....................................... 9-52
  OracleDate Properties ........................................................... 9-56
  OracleDate Methods .............................................................. 9-60

OracleDecimal Structure ............................................................ 9-65
  OracleDecimal Members ........................................................ 9-67
  OracleDecimal Constructors ................................................... 9-72
  OracleDecimal Static Fields ................................................... 9-78
  OracleDecimal Static (Comparison) Methods ......................... 9-82
  OracleDecimal Static (Manipulation) Methods ......................... 9-87
  OracleDecimal Static (Logarithmic) Methods ......................... 9-101
  OracleDecimal Static (Trigonometric) Methods ....................... 9-106
  OracleDecimal Static (Comparison) Operators ....................... 9-112
  OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal) ........ 9-120
  OracleDecimal Static Operators (Conversion from OracleDecimal to .NET) .................. 9-124
  OracleDecimal Properties ..................................................... 9-129
  OracleDecimal Instance Methods ........................................... 9-133

OracleIntervalDS Structure .......................................................... 9-139
  OracleIntervalDS Members ..................................................... 9-141
  OracleIntervalDS Constructors .............................................. 9-144
  OracleIntervalDS Static Fields .............................................. 9-149
  OracleIntervalDS Static Methods .......................................... 9-151
  OracleIntervalDS Static Operators ........................................ 9-157
  OracleIntervalDS Type Conversions ...................................... 9-165
  OracleIntervalDS Properties ................................................ 9-168
  OracleIntervalDS Methods .................................................... 9-173

OracleIntervalYM Structure .......................................................... 9-176
  OracleIntervalYM Members .................................................... 9-178
  OracleIntervalYM Constructors .............................................. 9-181
  OracleIntervalYM Static Fields .............................................. 9-185
OracleTypeException Class ................................................................. 10-2
OracleTypeException Members .................................................. 10-3
OracleTypeException Constructors ........................................ 10-5
OracleTypeException Static Methods ....................................... 10-7
Send Us Your Comments

Oracle Data Provider for .NET Developer's Guide 10g Release 1 (10.1.0.3)
Part No. B14164-01

Oracle 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) 506-7357. Attn: Oracle Database for Windows Documentation
- Postal service:
  Oracle Corporation
  Oracle Database for Windows Documentation Manager
  500 Oracle Parkway, Mailstop 1op4
  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.
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
- Documentation Accessibility
- Structure
- Related Documents
- Conventions

**Audience**

*Oracle Data Provider for .NET Developer’s Guide* is intended for programmers 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 programming 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.

**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.

Structure

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 Oracle Data Provider for .NET"
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 DB.

Chapter 4, "Oracle Data Provider for .NET Classes"
Describes the classes and public methods Oracle Data Provider for .NET exposes for ADO.NET programmers.

Chapter 5, "Oracle Data Provider for .NET XML-Related Classes"
Describes Oracle Data Provider for .NET XML-related classes and enumerations.

Chapter 6, "Oracle Data Provider for .NET Globalization Classes"
Describes the Oracle Data Provider for .NET globalization classes.

Chapter 7, "Oracle Data Provider for .NET Failover Classes"
Describes the Oracle Data Provider for .NET failover classes and enumerations.

Chapter 8, "Oracle Data Provider for .NET Types Classes"
Describes the large object and REF CURSOR objects provided by Oracle Data Provider for .NET.

Chapter 9, "Oracle Data Provider for .NET Types Structures"
Describes the types structures and objects provided by Oracle Data Provider for .NET.
Chapter 10, "Oracle Data Provider for .NET Types Exceptions"
Describes the Oracle Data Provider for .NET Types exceptions.

Glossary
Defines terms used in this document.

Related Documents
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 Call Interface Programmer’s 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
- Oracle Database Security Guide

Many of the examples in this book use the sample schemas, which are installed by default when you select the Basic Installation option with an Oracle Database installation. 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
and
http://msdn.microsoft.com/library

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.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong></td>
<td>Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.</td>
<td>When you specify this clause, you create an <strong>index-organized table</strong>.</td>
</tr>
<tr>
<td><strong>Italics</strong></td>
<td>Italic typeface indicates book titles or emphasis.</td>
<td><strong>Oracle Database Concepts</strong></td>
</tr>
<tr>
<td>UPPERCASE monospace (fixed-width) font</td>
<td>Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, Recovery Manager 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.</td>
<td>You can specify this clause only for a <strong>NUMBER</strong> column. You can back up the database by using the <strong>BACKUP</strong> command. Query the <strong>TABLE_NAME</strong> column in the <strong>USER_TABLES</strong> data dictionary view. Use the <strong>DBMS_STATS.GENERATE_STATS</strong> procedure.</td>
</tr>
<tr>
<td>lowercase monospace (fixed-width) font</td>
<td>Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names and connect identifiers, 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.</td>
<td>Enter sqlplus to start SQL*Plus. The password is specified in the <strong>orapwd</strong> file. Back up the datafiles and control files in the /disk1/oracle/dbs directory. The <strong>department_id</strong>, <strong>department_name</strong>, and <strong>location_id</strong> columns are in the <strong>hr.departments</strong> table. Set the <strong>QUERY_REWRITE_ENABLED</strong> initialization parameter to <strong>true</strong>. Connect as oe user. The <strong>JRepUtil</strong> class implements these methods.</td>
</tr>
<tr>
<td>lowercase italic monospace (fixed-width) font</td>
<td>Lowercase italic monospace font represents placeholders or variables.</td>
<td>You can specify the <strong>parallel_clause</strong>. Run <strong>old_release.SQL</strong> where <strong>old_release</strong> refers to the release you installed prior to upgrading.</td>
</tr>
</tbody>
</table>
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:

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

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

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>Anything enclosed in brackets is optional.</td>
<td>DECIMAL (digits [, precision])</td>
</tr>
<tr>
<td>{ }</td>
<td>Braces are used for grouping items.</td>
<td>{ENABLE</td>
</tr>
<tr>
<td></td>
<td>A vertical bar represents a choice of two options.</td>
<td>{ENABLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[COMPRESS</td>
</tr>
<tr>
<td>...</td>
<td>Ellipses points mean repetition in syntax descriptions.</td>
<td>CREATE TABLE ... AS subquery;</td>
</tr>
<tr>
<td></td>
<td>In addition, ellipses points can mean an omission in code examples or text.</td>
<td>SELECT col1, col2, ..., coln FROM employees;</td>
</tr>
<tr>
<td>Other symbols</td>
<td>You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (</td>
<td>), and ellipses points (...) exactly as shown.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>acct CONSTANT NUMBER(4) := 3;</td>
</tr>
<tr>
<td>Italics</td>
<td>Italicized text indicates placeholders or variables for which you must supply particular values.</td>
<td>CONNECT SYSTEM/system_password</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB_NAME = database_name</td>
</tr>
<tr>
<td>UPPERCASE</td>
<td>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. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.</td>
<td>SELECT last_name, employee_id FROM employees;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELECT * FROM USER_TABLES;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DROP TABLE hr.employees;</td>
</tr>
<tr>
<td>lowercase</td>
<td>Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.</td>
<td>SELECT last_name, employee_id FROM employees;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sqlplus hr/hr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CREATE USER mjones IDENTIFIED BY ty3MU9;</td>
</tr>
</tbody>
</table>

Conventions for Windows Operating Systems

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

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose Start &gt; menu item</td>
<td>How to start a program.</td>
<td>To start the Database Configuration Assistant, choose Start &gt; Programs &gt; Oracle - HOME_NAME &gt; Configuration and Migration Tools &gt; Database Configuration Assistant.</td>
</tr>
<tr>
<td>Convention</td>
<td>Meaning</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>File and directory</td>
<td>File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (&lt;), right angle bracket (&gt;), colon (:), double quotation marks (&quot;), slash (/), pipe (</td>
<td>), and dash (-). The special character backslash () is treated as an element separator, even when it appears in quotes. If the filename begins with \, then Windows assumes it uses the Universal Naming Convention.</td>
</tr>
<tr>
<td>names</td>
<td>C:/&gt; 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.</td>
<td>C:\oracle\oradata&gt;</td>
</tr>
<tr>
<td>Special characters</td>
<td>The backslash () special character is sometimes required as an escape character for the double quotation mark (&quot;) 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.</td>
<td>C:&gt; exp HR/HR TABLES=emp QUERY=&quot;WHERE job='REP'&quot;</td>
</tr>
<tr>
<td>HOME_NAME</td>
<td>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.</td>
<td>C:&gt; net start OracleHOME_NAME=TNSTNSListener</td>
</tr>
<tr>
<td>Convention</td>
<td>Meaning</td>
<td>Example</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ORACLE_HOME and</td>
<td>In releases prior to Oracle8i release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level ORACLE_HOME directory. The default for Windows NT 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\product\10.1.0. 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\product\10.1.0\db_n, where n is the latest Oracle home 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 Installation Guide for Windows for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.</td>
<td>Go to the ORACLE_BASE\ORACLE_HOME\rdbms\admin directory.</td>
</tr>
</tbody>
</table>
What's New in Oracle Data Provider for .NET?

This section describes new features in Oracle Data Provider for .NET 10g Release 1 (10.1.0.3) 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 10.1.0.3
- 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 10.1.0.3

Oracle Data Provider for .NET release 10.1.0.3 includes the following:

- Statement Caching
  This feature provides and manages a cache of statements for each session. The developer can control which statements are cached and how many. This improves performance and scalability.

  See Also: "Statement Caching" on page 3-21

- .NET Framework 1.1 Enhancements
  These enhancements expose new ADO.NET functionality that was introduced in Microsoft .NET Framework 1.1.

  See Also:
  - "EnlistDistributedTransaction" on page 4-76
  - "HasRows" on page 4-118

- Support for Command Cancellation
  These two new features relate to command cancellation. The CommandTimeout feature cancels the execution of a command when a specified amount of time elapses after the execution, while the Cancel method can be called explicitly by the application to terminate the execution of a command.
DerivedParameters Method
This method populates the parameter collection for the OracleCommand that represents a stored procedure or function by querying the database for the parameter information.

See Also: "DerivedParameters" on page 4-44

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 3-11

■ 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 3-35
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 3-15

Support for InitialLOBFetchSize property on OracleCommand and OracleDataReader objects

See Also: "Obtaining LOB Data" on page 3-27
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)**
- **ODP.NET Assembly**
- **Using ODP.NET in a Simple Application**

### 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](https://docs.microsoft.com/en-us/dotnet/api/). 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 Namespace

The `Oracle.DataAccess.Client` contains ODP.NET classes and enumerations.

### Oracle.DataAccess.Client Classes

*Table 1–1* lists the client classes.
### Table 1–1  Oracle.DataAccess.Client Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleCommand Class</td>
<td>An OracleCommand object represents a SQL command, a stored procedure or function, or a table name</td>
</tr>
<tr>
<td>OracleCommandBuilder Class</td>
<td>An OracleCommandBuilder object provides automatic SQL generation for the OracleDataAdapter when updates are made to the database</td>
</tr>
<tr>
<td>OracleConnection Class</td>
<td>An OracleConnection object represents a connection to an Oracle database</td>
</tr>
<tr>
<td>OracleDataAdapter Class</td>
<td>An OracleDataAdapter object represents a data provider object that communicates with the DataSet</td>
</tr>
<tr>
<td>OracleDataReader Class</td>
<td>An OracleDataReader object represents a forward-only, read-only, in-memory result set</td>
</tr>
<tr>
<td>OracleError Class</td>
<td>The OracleError object represents an error reported by an Oracle database</td>
</tr>
<tr>
<td>OracleErrorCollection Class</td>
<td>An OracleErrorCollection object represents a collection of OracleErrors</td>
</tr>
<tr>
<td>OracleException Class</td>
<td>The OracleException object represents an exception that is thrown when Oracle Data Provider for .NET encounters an error</td>
</tr>
<tr>
<td>OracleFailoverEventArgs Class</td>
<td>The OracleFailoverEventArgs object provides event data for the OracleConnection.Failover event</td>
</tr>
<tr>
<td>OracleFailoverEventHandler Delegate</td>
<td>The OracleFailoverEventHandler delegate represents the signature of the method that handles the OracleConnection.Failover event</td>
</tr>
<tr>
<td>OracleGlobalization Class</td>
<td>The OracleGlobalization class is used to obtain and set the Oracle globalization settings of the session, thread, and local computer (read-only)</td>
</tr>
<tr>
<td>OracleInfoMessageEventArgs Class</td>
<td>The OracleInfoMessageEventArgs object provides event data for the OracleConnection.InfoMessage event</td>
</tr>
<tr>
<td>OracleInfoMessageEventHandler Delegate</td>
<td>The OracleInfoMessageEventHandler delegate represents the signature of the method that handles the OracleConnection.InfoMessage event</td>
</tr>
<tr>
<td>OracleParameter Class</td>
<td>An OracleParameter object represents a parameter for an OracleCommand</td>
</tr>
<tr>
<td>OracleParameterCollection Class</td>
<td>An OracleParameterCollection object represents a collection of OracleParameters</td>
</tr>
<tr>
<td>OracleRowUpdatedEventArgs Class</td>
<td>The OracleRowUpdatedEventArgs object provides event data for the OracleDataAdapter.RowUpdated event</td>
</tr>
<tr>
<td>OracleRowUpdatedEventHandler Delegate</td>
<td>The OracleRowUpdatedEventHandler delegate represents the signature of the method that handles the OracleDataAdapter.RowUpdated event</td>
</tr>
</tbody>
</table>
Table 1–2 lists the client enumerations.

Table 1–2  Oracle.DataAccess.Client Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FailoverEvent Enumeration</td>
<td>FailoverEvent enumerated values are used to explicitly specify the state of the failover</td>
</tr>
<tr>
<td>FailoverReturnCode</td>
<td>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</td>
</tr>
<tr>
<td>FailoverType Enumeration</td>
<td>FailoverType enumerated values are used to indicate the type of failover event that was raised</td>
</tr>
<tr>
<td>OracleCollectionType</td>
<td>OracleCollectionType enumerated values specify whether the OracleParameter object represents a collection, and if so, specifies the collection type</td>
</tr>
<tr>
<td>OracleDbType</td>
<td>OracleDbType enumerated values are used to explicitly specify the OracleDbType of an OracleParameter</td>
</tr>
<tr>
<td>OracleParameterStatus</td>
<td>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</td>
</tr>
<tr>
<td>OracleXmlCommandType</td>
<td>The OracleXmlCommandType enumeration specifies the values that are allowed for the OracleXmlCommandType property of the OracleCommand class</td>
</tr>
</tbody>
</table>
Oracle.DataAccess.Types Namespace

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

Oracle.DataAccess.Types Structures

Table 1–3 lists the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleBinary Structure</td>
<td>The OracleBinary structure represents a variable-length stream of binary data</td>
</tr>
<tr>
<td>OracleDate Structure</td>
<td>The OracleDate structure represents the Oracle DATE datatype</td>
</tr>
<tr>
<td>OracleDecimal Structure</td>
<td>The OracleDecimal structure represents an Oracle NUMBER in the database or any Oracle numeric value</td>
</tr>
<tr>
<td>OracleIntervalDS Structure</td>
<td>The OracleIntervalDS structure represents the Oracle INTERVAL DAY TO SECOND datatype</td>
</tr>
<tr>
<td>OracleIntervalYM Structure</td>
<td>The OracleIntervalYM structure represents the Oracle INTERVAL YEAR TO MONTH datatype</td>
</tr>
<tr>
<td>OracleString Structure</td>
<td>The OracleString structure represents a variable-length stream of characters</td>
</tr>
<tr>
<td>OracleTimeStamp Structure</td>
<td>The OracleTimeStamp structure represents the Oracle Timestamp datatype</td>
</tr>
<tr>
<td>OracleTimeStampLTZ Structure</td>
<td>The OracleTimeStampLTZ structure represents the Oracle TIMESTAMP WITH LOCAL TIME ZONE data type</td>
</tr>
<tr>
<td>OracleTimeStampTZ Structure</td>
<td>The OracleTimeStampTZ structure represents the Oracle TIMESTAMP WITH TIME ZONE data type</td>
</tr>
</tbody>
</table>

Oracle.DataAccess.Types Exceptions

Types exceptions are thrown only by ODP.NET types structures. Table 1–4 lists the exceptions.

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleTypeException Class</td>
<td>The OracleTypeException object is the base exception class for handling exceptions that occur in the ODP.NET Type classes</td>
</tr>
<tr>
<td>OracleNullValueException Class</td>
<td>The OracleNullValueException represents an exception that is thrown when trying to access an ODP.NET Type structure that is null</td>
</tr>
<tr>
<td>OracleTruncateException Class</td>
<td>The OracleTruncateException class represents an exception that is thrown when truncation in an ODP.NET Type class occurs</td>
</tr>
</tbody>
</table>

Oracle.DataAccess.Types Classes

Table 1–5 lists the classes.
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.

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

class Sample
{
    static void Main()
    {
        // Connect to Oracle
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Display Version Number
        Console.WriteLine("Connected to Oracle " + con.ServerVersion);

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

Table 1–5  `Oracle.DataAccess.Types` Classes

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

Note: Additional samples are provided in the `ORACLE_BASE\ORACLE_HOME\ODP.NET\Samples` directory.
This chapter describes installation and configuration requirements for Oracle Data Provider for .NET.

This chapter contains these topics:

- System Requirements
- Installing Oracle Data Provider for .NET
- File Locations

System Requirements

Oracle Data Provider for .NET requires the following:

- Microsoft .NET Framework 1.0 or higher.
- Access to Oracle8i 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 Oracle9i Database release 2 (9.2) or higher.
- Applications using OracleXmlStream and OracleXmlType classes with schema-based XMLType require Oracle Database 10g.
- 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
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 Dynamic Help is registered with Visual Studio .NET, providing context-sensitive online help that is seamlessly integrated with Visual Studio .NET Dynamic Help. With Dynamic Help, the user can access ODP.NET documentation within the Visual Studio .NET IDE by placing the cursor on an ODP.NET keyword and pressing the F1 function key.

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.

See Also: http://msdn.microsoft.com/netframework
http://otn.oracle.com/tech/xml/xdkhome.html to download the Oracle XML Developer’s Kit (XDK)
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
- OracleCommand Object
- ODP.NET Types Overview
- Obtaining Data From an OracleDataReader
- PL/SQL REF CURSOR and OracleRefCursor
- LOB Support
- ODP.NET XML Support
- OracleDataAdapter Safe Type Mapping
- OracleDataAdapter Requery Property
- Guaranteeing Uniqueness in Updating DataSet to Database
- Globalization Support
- Debug Tracing

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

```csharp
// C#
using System;
using Oracle.DataAccess.Client;

class ConnectionSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        //using connection string attributes to connect to Oracle Database
```
con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle";
con.Open();
Console.WriteLine("Connected to Oracle" + con.ServerVersion);

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
Console.WriteLine("Disconnected");
}
}

See Also:

- "OracleConnection Properties" on page 4-62 for detailed information on connection attributes
- "OracleCommand Object" on page 3-10 for detailed information on statement caching

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
- Validate Connection

Connection Pooling Example

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

// C#

using System;
using Oracle.DataAccess.Client;

class ConnectionPoolingSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        //Open a connection using ConnectionString attributes related to connection pooling.
        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;" +
            "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
            "Max Pool Size=20;Incr Pool Size=10;Decr Pool Size=5;Validate Connection=true;";
"Incr Pool Size=5; Decr Pool Size=2";
con.Open();
Console.WriteLine("Connection pool successfully created");

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
Console.WriteLine("Connection is placed back into the pool.");

Using Connection Pooling

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 pooling and returning connections to the application.

The connection pooling service creates connection pools by 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 Validate Connection attribute validates connections coming out of the pool. This attribute should only be used when absolutely necessary because it causes a server round-trip to the database to validate each connection right before it is provided to the application. If invalid connections are uncommon, developers can create their own event handler to retrieve a new connection, rather than using Validate Connection. This generally provides better performance.

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.

/* Create an OS-authenticated user in the database
Assume init.ora does not have OS_AUTHENT_PREFIX set &
a local OS user called 'FooUser' exists with the password 'FooUser':

    create user OPS$FooUser identified by FooUser;
    grant connect, resource to OPS$FooUser;

Login as FooUsers and execute the sample
See documentation for details on how to configure an OS-Authenticated user */

// C#
using System;
using Oracle.DataAccess.Client;

class OSAuthenticationSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();
        //Establish connection using OS Authentication
        con.ConnectionString = "User Id=/;Data Source=oracle;";
        con.Open();
        Console.WriteLine("Connected to Oracle" + con.ServerVersion);
        // Close and Dispose OracleConnection object
        con.Close();
        con.Dispose();
        Console.WriteLine("Disconnected");
    }
}

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#
using System;
using Oracle.DataAccess.Client;
class PrivilegedConnectionSample
{
static void Main()
{
OracleConnection con = new OracleConnection();

//Connect SYS/SYS as SYSDBA
con.ConnectionString = "User Id=SYS;Password=SYS;" +
"DBA Privilege=SYSDBA;Data Source=oracle;";
con.Open();
Console.WriteLine("Connected to Oracle" + con.ServerVersion);

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
Console.WriteLine("Disconnected");
}
}

See Also: "DBA Privilege" on page 4-64 for further information on privileged connections in the database

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:

/* Database Setup
connect / as sysdba;
drop user testexpire cascade;
-- create user "testexpire" with password "testexpire"
grant connect, resource to testexpire identified by testexpire;
alter user testexpire password expire;
*/

// C#
using System;
using Oracle.DataAccess.Client;

class PasswordExpirationSample
{
static void Main()
{
OracleConnection con = new OracleConnection();

try
{
con.ConnectionString = "User Id=testexpire;Password=testexpire;Data Source=oracle;";
con.Open();
Console.WriteLine("Connected to Oracle" + con.ServerVersion);
}
catch (OracleException ex)
{
Console.WriteLine(ex.Message);
}
// check the error number
// ORA-28001: the password has expired
if (ex.Number == 28001)
{
    Console.WriteLine("Changing password to panther");
    con.OpenWithNewPassword("panther");
    Console.WriteLine("Connected with new password.");
}
}
finally
{
    // Close and Dispose OracleConnection object
    con.Close();
    con.Dispose();
    Console.WriteLine("Disconnected");
}
}

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

**See Also:** "OpenWithNewPassword" on page 4-80 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 (user id and password) 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.

```csharp
/* Create a proxy user and modified scott to allow proxy connection:
    create user appserver identified by eagle;
    grant connect, resource to appserver;
    alter user scott grant connect through appserver;
*/

// C#

using System;
using Oracle.DataAccess.Client;

class ProxyAuthenticationSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        // Connecting using proxy authentication
```
Connecting to the Oracle Database

```csharp
con.ConnectionString = "User Id=scott;Password=tiger;" +
    "Data Source=oracle;Proxy User Id=appserver;Proxy Password=eagle; ";
con.Open();
Console.WriteLine("Connected to Oracle" + con.ServerVersion);

// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
Console.WriteLine("Disconnected");
```

**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.

**TAF Notification**

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 Failover Occurs**

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.

### Registering an Event Handler for Failover

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

```csharp
// C#
using System;
using Oracle.DataAccess.Client;

class TAPCallBackSample
{
    public static FailoverReturnCode OnFailover(object sender, OracleFailoverEventArgs eventArgs)
    {
        switch (eventArgs.FailoverEvent)
        {
            case FailoverEvent.Begin:
                Console.WriteLine(" \nFailover Begin - Failing Over ... Please standby \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");
                return FailoverReturnCode.Retry;
            default:
                Console.WriteLine("Bad Failover Event: %d.\n", eventArgs.FailoverEvent);
                break;
        }
        return FailoverReturnCode.Success;
    }

    static void Main()
    {
        OracleConnection con = new OracleConnection();

        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();
        con.Failover += new OracleFailoverEventHandler(OnFailover);
        Console.WriteLine("Event Handler is successfully registered");
    }
} /* 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 7-8
- "OracleFailoverEventArgs Class" on page 7-2

---

### 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**
- **Statement Caching**

---

### 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 4-282

Datatypes BINARY_FLOAT and BINARY_DOUBLE

Starting from Oracle Database 10g, 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 4-130
- "GetFloat" on page 4-131

OracleDbType Enumeration Type

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

Table 3–2 lists all the OracleDbType enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFile</td>
<td>Oracle BFILE type</td>
</tr>
<tr>
<td>Blob</td>
<td>Oracle BLOB type</td>
</tr>
<tr>
<td>Byte</td>
<td>byte type</td>
</tr>
<tr>
<td>Char</td>
<td>Oracle CHAR type</td>
</tr>
</tbody>
</table>
Table 3–2 (Cont.) OracleDbType Enumeration Values

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clob</td>
<td>Oracle CLOB type</td>
</tr>
<tr>
<td>Date</td>
<td>Oracle DATE type</td>
</tr>
<tr>
<td>Decimal</td>
<td>Oracle NUMBER type</td>
</tr>
<tr>
<td>Double</td>
<td>8-byte FLOAT type</td>
</tr>
<tr>
<td>Int16</td>
<td>2-byte INTEGER type</td>
</tr>
<tr>
<td>Int32</td>
<td>4-byte INTEGER type</td>
</tr>
<tr>
<td>Int64</td>
<td>8-byte INTEGER type</td>
</tr>
<tr>
<td>IntervalDS</td>
<td>Oracle INTERVAL DAY TO SECOND type</td>
</tr>
<tr>
<td>IntervalYM</td>
<td>Oracle INTERVAL YEAR TO MONTH type</td>
</tr>
<tr>
<td>Long</td>
<td>Oracle LONG type</td>
</tr>
<tr>
<td>LongRaw</td>
<td>Oracle LONG RAW type</td>
</tr>
<tr>
<td>NChar</td>
<td>Oracle NCHAR type</td>
</tr>
<tr>
<td>NClob</td>
<td>Oracle NCLOB type</td>
</tr>
<tr>
<td>NVarchar2</td>
<td>Oracle NVARCHAR2 type</td>
</tr>
<tr>
<td>Raw</td>
<td>Oracle RAW type</td>
</tr>
<tr>
<td>RefCursor</td>
<td>Oracle REF CURSOR type</td>
</tr>
<tr>
<td>Single</td>
<td>4-byte FLOAT type</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>Oracle TIMESTAMP type</td>
</tr>
<tr>
<td>TimeStampLTZ</td>
<td>Oracle TIMESTAMP WITH LOCAL TIME ZONE type</td>
</tr>
<tr>
<td>TimeStampTZ</td>
<td>Oracle TIMESTAMP WITH TIME ZONE type</td>
</tr>
<tr>
<td>Varchar2</td>
<td>Oracle VARCHAR2 type</td>
</tr>
<tr>
<td>XmlType</td>
<td>Oracle XMLType type</td>
</tr>
</tbody>
</table>

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 3–3.

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

<table>
<thead>
<tr>
<th>OracleDbType</th>
<th>System.Data.DbType</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFile</td>
<td>Object</td>
</tr>
<tr>
<td>Blob</td>
<td>Object</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte</td>
</tr>
<tr>
<td>Char</td>
<td>StringFixedLength</td>
</tr>
<tr>
<td>Clob</td>
<td>Object</td>
</tr>
</tbody>
</table>
Inference of OracleDbType from DbType

In the `OracleParameter` class, specifying the value of `DbType` infers the value of `OracleDbType` as shown in Table 3–4.

**Table 3–4  Inference of OracleDbType from DbType**

<table>
<thead>
<tr>
<th>OracleDbType</th>
<th>System.Data.DbType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Date</td>
</tr>
<tr>
<td>Decimal</td>
<td>Decimal</td>
</tr>
<tr>
<td>Double</td>
<td>Double</td>
</tr>
<tr>
<td>Int16</td>
<td>Int16</td>
</tr>
<tr>
<td>Int32</td>
<td>Int32</td>
</tr>
<tr>
<td>Int64</td>
<td>Int64</td>
</tr>
<tr>
<td>IntervalDS</td>
<td>(TimeSpan)</td>
</tr>
<tr>
<td>IntervalYM</td>
<td>Int64</td>
</tr>
<tr>
<td>Long</td>
<td>String</td>
</tr>
<tr>
<td>LongRaw</td>
<td>Binary</td>
</tr>
<tr>
<td>NChar</td>
<td>StringFixedLength</td>
</tr>
<tr>
<td>NClob</td>
<td>Object</td>
</tr>
<tr>
<td>NVarchar2</td>
<td>String</td>
</tr>
<tr>
<td>Raw</td>
<td>Binary</td>
</tr>
<tr>
<td>RefCursor</td>
<td>Object</td>
</tr>
<tr>
<td>Single</td>
<td>Single</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>DateTime</td>
</tr>
<tr>
<td>TimeStampLTZ</td>
<td>DateTime</td>
</tr>
<tr>
<td>TimeStampTZ</td>
<td>DateTime</td>
</tr>
<tr>
<td>Varchar2</td>
<td>String</td>
</tr>
<tr>
<td>XmlType</td>
<td>String</td>
</tr>
</tbody>
</table>

Inference of OracleDbType from DbType

In the `OracleParameter` class, specifying the value of `DbType` infers the value of `OracleDbType` as shown in Table 3–4.

**Table 3–4  Inference of OracleDbType from DbType**

<table>
<thead>
<tr>
<th>System.Data.DbType</th>
<th>OracleDbType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>Raw</td>
</tr>
<tr>
<td>Boolean</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Byte</td>
<td>Byte</td>
</tr>
<tr>
<td>Currency</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Date</td>
<td>Date</td>
</tr>
<tr>
<td>DateTime</td>
<td>TimeStamp</td>
</tr>
<tr>
<td>Decimal</td>
<td>Decimal</td>
</tr>
<tr>
<td>Double</td>
<td>Double</td>
</tr>
<tr>
<td>Guid</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Int16</td>
<td>Int16</td>
</tr>
<tr>
<td>Int32</td>
<td>Int32</td>
</tr>
</tbody>
</table>
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 3–5 shows the inference of DbType and OracleDbType from Value when the type of Value is one of the .NET Framework datatypes.

<table>
<thead>
<tr>
<th>Value (.NET Datatypes)</th>
<th>System.Data.DbType</th>
<th>OracleDbType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Byte</td>
<td>Byte</td>
</tr>
<tr>
<td>Byte[]</td>
<td>Binary</td>
<td>Raw</td>
</tr>
<tr>
<td>Char / Char []</td>
<td>String</td>
<td>Varchar2</td>
</tr>
<tr>
<td>DateTime</td>
<td>DateTime</td>
<td>TimeStamp</td>
</tr>
<tr>
<td>Decimal</td>
<td>Decimal</td>
<td>Decimal</td>
</tr>
<tr>
<td>Double</td>
<td>Double</td>
<td>Double</td>
</tr>
<tr>
<td>Float</td>
<td>Single</td>
<td>Single</td>
</tr>
<tr>
<td>Int16</td>
<td>Int16</td>
<td>Int16</td>
</tr>
<tr>
<td>Int32</td>
<td>Int32</td>
<td>Int32</td>
</tr>
<tr>
<td>Int64</td>
<td>Int64</td>
<td>Int64</td>
</tr>
<tr>
<td>Single</td>
<td>Single</td>
<td>Single</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
<td>Varchar2</td>
</tr>
<tr>
<td>TimeSpan</td>
<td>TimeSpan</td>
<td>IntervalDS</td>
</tr>
</tbody>
</table>

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 3–6 shows the inference of DbType and OracleDbType from Value when type of Value is one of Oracle.DataAccess.Types.

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

**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: MYPACK**

```plsql
/* Setup the tables and required PL/SQL: */
CREATE TABLE T1(COL1 number, COL2 varchar2(20));

CREATE or replace 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;

CREATE or REPLACE 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 Param2 to Param1
        Param1(1) := Param2(1);
        Param1(2) := NULL;
        Param1(3) := Param2(3);
        -- copy all elements from Param1 to Param2
        Param2(1) := Param1(1);
        Param2(2) := Param1(2);
        Param2(3) := Param1(3);
        -- insert some values to db
        FOR i IN 1..3 LOOP
            insert into T1 values(i,Param2(i));
        END LOOP;
    END TestVarchar2;
END MYPACK;
*/

// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class AssociativeArraySample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();
        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle";
        con.Open();
    }
}
```

---

**Oracle Data Provider for .NET Developer's Guide**

3-16

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

OracleParameter Param1 = cmd.Parameters.Add("1", OracleDbType.Varchar2);
OracleParameter Param2 = cmd.Parameters.Add("2", OracleDbType.Varchar2);
OracleParameter Param3 = cmd.Parameters.Add("3", OracleDbType.Varchar2);

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
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

// Setup the ArrayBindStatus for Param1

// Setup the ArrayBindSize for Param2

// Setup the ArrayBindSize for Param3

// execute the cmd
cmd.ExecuteNonQuery();

//print out the parameter's values
Console.WriteLine("parameter values after executing the PL/SQL block");
for (int i = 0; i < 3; i++)
    Console.WriteLine("Param2[{0}] = {1} ", i,
        (cmd.Parameters[1].Value as Array).GetValue(i));

for (int i = 0; i < 3; i++)
    Console.WriteLine("Param3[{0}] = {1} ", i,
        (cmd.Parameters[2].Value as Array).GetValue(i));

// Close and Dispose OracleConnection object
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.

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

class ArrayBindSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();
        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();
        Console.WriteLine("Connected successfully");

        int[] myArrayDeptNo = new int[3] { 10, 20, 30 };
        OracleCommand cmd = new OracleCommand();

        // Set the command text on an OracleCommand object
        cmd.CommandText = "insert into dept(deptno) values (:deptno)";
        cmd.Connection = con;

        // 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();
        Console.WriteLine("Insert Completed Successfully");

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

```csharp
/* Database Setup
drop table depttest;
create table depttest(deptno number(2));
*/

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;
```
class ArrayBindExceptionSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();
        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();

        OracleCommand cmd = new OracleCommand();

        // Start a transaction
        OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

        try
        {
            int[] myArrayDeptNo = new int[] { 10, 200000, 30 };
            // int[] myArrayDeptNo = new int[]{10, 20, 30};

            // Set the command text on an OracleCommand object
            cmd.CommandText = "insert into depttest(deptno) values (:deptno)";
            cmd.Connection = con;

            // 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();
        }
        catch (OracleException e)
        {
            Console.WriteLine("OracleException {0} occured", e.Message);
            if (e.Number == 24381)
            {
                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);
                }
            }
            txn.Commit();
        }
        cmd.Parameters.Clear();
        cmd.CommandText = "select count(*) from depttest";

        decimal rows = (decimal)cmd.ExecuteScalar();

        Console.WriteLine("{0} row have been inserted", rows);
        con.Close();
        con.Dispose();
    }
}

See Also: "ArrayBindIndex" on page 4-166 for more information
**OracleParameterStatus Enumeration Types** Table 3–7 provides different values for OracleParameterStatus enumeration.

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

**Statement Caching**

Statement caching eliminates the need to parse each SQL or PL/SQL statement before the execution by caching server cursors created from its initial execution. Subsequent executions of the same statement (possibly with different parameter values) can reuse the parsed information from the cursor and execute the statement without parsing for better performance.

It is recommended that only those statements that will be repeatedly executed be cached to see performance gains from statement caching. Furthermore, SQL or PL/SQL statements should use parameters rather than using literal values. This is because when statement caching is enabled, parameterized statements that have already been prepared can be re-used for subsequent executions of the same statement with different parameter values.

**Statement Caching Connection String Attributes**

The following connection string attributes control the behavior of the ODP.NET statement caching feature:

- **Statement Cache Size**
  
  This attribute enables or disables ODP.NET Statement Caching. By default, this attribute is set to 0 and ODP.NET Statement Caching is disabled. If is set to a value greater than 0, ODP.NET Statement Caching is enabled and the value specified is used as maximum number of statements that can be cached for a connection. Once a connection has cached up to the specified maximum cache size, the least recently used cursor is freed to make room to cache the newly created cursor.

- **Statement Cache Purge**
  
  This attribute provides a way for connections to purge all statements that are cached when a connection is closed or placed back into the connection pool. By default, this attribute is set to False, which means that cursors are not freed when connections are placed back into the pool.

**Statement Caching Methods and Properties**

The following method and property are only relevant when Statement Caching is enabled:

- **OracleCommand.AddToStatementCache** property
If statement caching is enabled, having this property set to True adds statements to the cache when they are executed. If statement caching is disabled or if this property is set to False, the executed statement is not cached. By default, this property is set to True.

- OracleConnection.PurgeStatementCache method

This method purges all the cached statements by closing all open cursors on the server that are associated with the particular connection. Note that statement caching remains enabled after this call.

Connections and Statement Caching
Statement caching is managed separately for each connection. Therefore, executing the same statement on different connections requires parsing once for each connection and caching a separate cursor for each connection.

Pooling and Statement Caching
Pooling and statement caching can be used in conjunction. If connection pooling is enabled and Statement Change Purge is set to False, statements executed on each separate connection are cached throughout the lifetime of the pooled connection.

If Statement Cache Purge is set to True, all the cached cursors are freed when the connection is placed back into the pool. When connection pooling is disabled, cursors are cached during the lifetime of the connection, but the cursors are closed when the OracleConnection object is closed or disposed.

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 3–8 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>
<thead>
<tr>
<th>Oracle Native Type</th>
<th>ODP.NET Type</th>
<th>.NET Framework Datatypes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFILE</td>
<td>OracleBFile class</td>
<td>System.Byte[]</td>
</tr>
<tr>
<td>BLOB</td>
<td>OracleBlob class</td>
<td>System.Byte[]</td>
</tr>
<tr>
<td>CHAR</td>
<td>OracleString structure</td>
<td>System.String</td>
</tr>
<tr>
<td>CLOB</td>
<td>OracleClob class</td>
<td>System.String</td>
</tr>
<tr>
<td>DATE</td>
<td>OracleDate structure</td>
<td>System.DateTime</td>
</tr>
<tr>
<td>INTERVAL DAY TO SECOND</td>
<td>OracleIntervalDS structure</td>
<td>System.TimeSpan</td>
</tr>
<tr>
<td>INTERVAL YEAR TO MONTH</td>
<td>OracleIntervalYM structure</td>
<td>System.Int64</td>
</tr>
<tr>
<td>LONG</td>
<td>OracleString structure</td>
<td>System.String</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>OracleBinary structure</td>
<td>System.Byte[]</td>
</tr>
</tbody>
</table>
Obtaining Data From anOracleDataReader

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 3–9 lists all the Oracle native database types that ODP.NET supports and the corresponding .NET Types that can represent the Oracle native type. If more than one .NET Type can be used to represent an Oracle native type, the first entry is the .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
- "OracleDataReader Class" on page 4-109

**Table 3–9 .NET Type Accessors**

<table>
<thead>
<tr>
<th>Oracle Native Datatype</th>
<th>.NET Type</th>
<th>Typed Accessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFILE</td>
<td>System.Byte[]</td>
<td>GetBytes</td>
</tr>
<tr>
<td>BINARY_DOUBLE</td>
<td>System.Double</td>
<td>GetDouble</td>
</tr>
<tr>
<td>BINARY_FLOAT</td>
<td>System.Single</td>
<td>GetFloat</td>
</tr>
<tr>
<td>BLOB</td>
<td>System.Byte[]</td>
<td>GetBytes</td>
</tr>
<tr>
<td>CHAR</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
<tr>
<td>CLOB</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
<tr>
<td>DATE</td>
<td>System.DateTime</td>
<td>GetDateTime</td>
</tr>
<tr>
<td>INTERVAL(DS)</td>
<td>System.Timespan</td>
<td>GetTimeSpan</td>
</tr>
<tr>
<td>INTERVAL (YM)</td>
<td>System.Int64</td>
<td>GetInt64</td>
</tr>
<tr>
<td>LONG</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>System.Byte[]</td>
<td>GetBytes</td>
</tr>
<tr>
<td>NCHAR</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
<tr>
<td>NCLOB</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
<tr>
<td>NUMBER</td>
<td>System.Decimal</td>
<td>GetDecimal</td>
</tr>
<tr>
<td></td>
<td>System.Byte</td>
<td>GetByte</td>
</tr>
<tr>
<td></td>
<td>System.Int16</td>
<td>GetInt16</td>
</tr>
<tr>
<td></td>
<td>System.Int32</td>
<td>GetInt32</td>
</tr>
<tr>
<td></td>
<td>System.Int64</td>
<td>GetInt64</td>
</tr>
<tr>
<td></td>
<td>System.Single</td>
<td>GetFloat</td>
</tr>
<tr>
<td></td>
<td>System.Double</td>
<td>GetDouble</td>
</tr>
<tr>
<td>NVARCHAR2</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
<tr>
<td>RAW</td>
<td>System.Byte[]</td>
<td>GetBytes</td>
</tr>
<tr>
<td>ROWID</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>System.DateTime</td>
<td>GetDateTime</td>
</tr>
<tr>
<td>TIMESTAMP WITH LOCAL TIME ZONE</td>
<td>System.DateTime</td>
<td>GetDateTime</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>System.DateTime</td>
<td>GetDateTime</td>
</tr>
<tr>
<td>UROWID</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
</tbody>
</table>
Certain methods and properties of the OracleDataReader require ODP.NET to map a NUMBER column to a .NET type based on the precision and scale of the column. These members are:

- Item Property
- GetFieldType Method
- GetValue Method
- GetValues Method

ODP.NET determines the appropriate .NET type by considering the following .NET types in order and selecting the first .NET type from the list that can represent the entire range of values of the column.

- System.Byte
- System.Int16
- System.Int32
- System.Int64
- System.Single
- System.Double
- System.Decimal

If no .NET type exists that can represent the entire range of values of the column, then an attempt is made to represent the column values as a System.Decimal. If the value in the column cannot be represented as a System.Decimal, then an exception is raised.

For example, consider two columns defined as NUMBER(4,0) and NUMBER(10,2). The first .NET types from the previous list that can represent the entire range of values of the columns are System.Int16 and System.Double, respectively. However, consider a column defined as NUMBER(20,10). In this case, there is no .NET type that can represent the entire range of values on the column, so an attempt is made to return values in the column as a System.Decimal. If a value in the column cannot be represented as a System.Decimal, then an exception is raised.

The Fill method of the OracleDataAdapter utilizes the OracleDataReader to populate or refresh a DataTable or DataSet with .NET types. As a result, the .NET type used to represent a NUMBER column in the DataTable or DataSet also depends on the precision and scale of the column.
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 3-22 for a list of all ODP.NET types

Table 3–10 lists the valid type accessors that ODP.NET uses to obtain ODP.NET Types for an Oracle native type.

Table 3–10  ODP.NET Type Accessors

<table>
<thead>
<tr>
<th>Oracle Native Database Type</th>
<th>ODP.NET Type</th>
<th>Typed Accessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFILE</td>
<td>OracleBFile</td>
<td>GetOracleBFile</td>
</tr>
<tr>
<td>BLOB</td>
<td>OracleBlob</td>
<td>GetOracleBlob</td>
</tr>
<tr>
<td></td>
<td>GetOracleBlobForUpdate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetOracleBinary</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>OracleString</td>
<td>GetOracleString</td>
</tr>
<tr>
<td>CLOB</td>
<td>OracleClob</td>
<td>GetOracleClob</td>
</tr>
<tr>
<td></td>
<td>GetOracleClobForUpdate</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>OracleDate</td>
<td>GetOracleDate</td>
</tr>
<tr>
<td>INTERVAL (DS)</td>
<td>OracleIntervalDS</td>
<td>GetOracleIntervalDS</td>
</tr>
<tr>
<td>INTERVAL (YM)</td>
<td>OracleIntervalYM</td>
<td>GetOracleIntervalYM</td>
</tr>
<tr>
<td>LONG</td>
<td>OracleString</td>
<td>GetOracleString</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>OracleBinary</td>
<td>GetOracleBinary</td>
</tr>
<tr>
<td>NCHAR</td>
<td>OracleString</td>
<td>GetOracleString</td>
</tr>
<tr>
<td>NCLOB</td>
<td>OracleString</td>
<td>GetOracleString</td>
</tr>
<tr>
<td>NUMBER</td>
<td>OracleDecimal</td>
<td>GetOracleDecimal</td>
</tr>
<tr>
<td>NVARCHAR2</td>
<td>OracleString</td>
<td>GetOracleString</td>
</tr>
<tr>
<td>RAW</td>
<td>OracleBinary</td>
<td>GetOracleBinary</td>
</tr>
<tr>
<td>ROWID</td>
<td>OracleString</td>
<td>GetOracleString</td>
</tr>
</tbody>
</table>
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 InitialLOBFetchSize property of the OracleCommand that created the OracleDataReader. By default, InitialLOBFetchSize is set to 0. If the InitialLOBFetchSize property value of

Table 3-10 (Cont.) ODP.NET Type Accessors

<table>
<thead>
<tr>
<th>Oracle Native Database Type</th>
<th>ODP.NET Type</th>
<th>Typed Accessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>OracleTimeStamp</td>
<td>GetOracleTime Stamp</td>
</tr>
<tr>
<td>TIMESTAMP WITH LOCAL TIME ZONE</td>
<td>OracleTimeStampLTZ</td>
<td>GetOracleTime StampLTZ</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>OracleTimeStampTZ</td>
<td>GetOracleTime StampTZ</td>
</tr>
<tr>
<td>UROWID</td>
<td>OracleString</td>
<td>GetOracleString</td>
</tr>
<tr>
<td>VARCHAR2</td>
<td>OracleString</td>
<td>GetOracleString</td>
</tr>
<tr>
<td>XMLType</td>
<td>OracleXmlType</td>
<td>GetOracleXmlType</td>
</tr>
</tbody>
</table>
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 property 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 3–11 and Table 3–12 list supported and not supported methods for the CLOB and BLOB datatypes when the OracleCommand InitialLOBFetchSize property is set to a nonzero value.

**Table 3–11  OracleDataReader CLOB Methods**

<table>
<thead>
<tr>
<th>Supported</th>
<th>Not Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetChars</td>
<td>GetOracleClob</td>
</tr>
<tr>
<td>GetString</td>
<td>GetOracleClobForUpdate</td>
</tr>
<tr>
<td>GetValue</td>
<td>GetOracleValue</td>
</tr>
<tr>
<td>GetValues</td>
<td>GetOracleValues</td>
</tr>
<tr>
<td>GetOracleString</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3–12  OracleDataReader BLOB Methods**

<table>
<thead>
<tr>
<th>Supported</th>
<th>Not Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBytes</td>
<td>GetOracleBlob</td>
</tr>
<tr>
<td>GetValue</td>
<td>GetOracleBlobForUpdate</td>
</tr>
<tr>
<td>GetValues</td>
<td>GetOracleValue</td>
</tr>
<tr>
<td>GetOracleBinary</td>
<td></td>
</tr>
<tr>
<td>GetOracleBinary</td>
<td></td>
</tr>
</tbody>
</table>

**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 3-32
- "InitialLOBFetchSize" on page 4-17
- "InitialLONGFetchSize" on page 4-18

**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.

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
PL/SQL REF CURSOR and OracleRefCursor

- 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. This section includes the following topics:

- Large Character and Large Binary Datatypes
- Oracle Data Provider for .NET LOB Objects
- Updating LOBs Using a DataSet
- Updating LOBs Using OracleCommand and OracleParameter
- Updating LOBs Using ODP.NET LOB Objects
- Temporary LOBs
Large Character and Large Binary 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 Oracle9i, but should not be used in new applications.

Oracle Data Provider for .NET LOB Objects

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**

<table>
<thead>
<tr>
<th>Oracle LOB Type</th>
<th>ODP.NET LOB object</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFILE</td>
<td>OracleBFile object</td>
</tr>
<tr>
<td>BLOB</td>
<td>OracleBlob object</td>
</tr>
<tr>
<td>CLOB</td>
<td>OracleClob object</td>
</tr>
<tr>
<td>NCLOB</td>
<td>OracleClob object</td>
</tr>
</tbody>
</table>

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 Oracle Database 10g 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 strings. 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 Oracle9i 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 OracleConnection 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.

ODP.NET XML Support

From Oracle8i 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 Oracle9i 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.
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:

  ```csharp
  // Create a table with an XMLType column in the database
  CREATE TABLE XML_TABLE(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:

```sql
// Create a table with an XMLType column in the database
CREATE TABLE XML_TABLE (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>
<thead>
<tr>
<th>Special Character</th>
<th>Meaning in XML</th>
<th>Entity Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Begins an XML tag</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>Ends an XML tag</td>
<td>&gt;</td>
</tr>
<tr>
<td>&quot;</td>
<td>Quotation mark</td>
<td>&quot;</td>
</tr>
<tr>
<td>'</td>
<td>Apostrophe or single quotation mark</td>
<td>'</td>
</tr>
<tr>
<td>&amp;</td>
<td>Ampersand</td>
<td>&amp;</td>
</tr>
</tbody>
</table>

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 3–15 lists the date and time format handling for different database releases.
Table 3–15  Database Release Date and Time Differences When Retrieving Data

<table>
<thead>
<tr>
<th>Database Release</th>
<th>Date and Time Format Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle8i release 3 (8.1.7) and Oracle9i release 1 (9.0.x)</td>
<td>Oracle DATE type data is always retrieved in the result XML document as the ISO Date and Time Format: <code>YYYY-MM-DDThh:mm:ss.sss</code> (ISO Format notation). The following string is the ISO Date and Time Format notation represented in the Oracle Date and Time Format notation: <code>YYYY-MM-DD&quot;T&quot;HH24:MI:SS.FF3</code>. TIMESTAMP and TIMESTAMPT WITH TIME ZONE are not supported for 8.1.7 and 9.0.x.</td>
</tr>
<tr>
<td>Oracle9i release 2 (9.2.x) and Oracle Database 10g</td>
<td>Oracle DATE type data is retrieved in the format specified using the NLS_DATE_FORMAT in the session. TIMESTAMP and TIMESTAMPT 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 back to the database, then all DATE and TIMESTAMP data must be retrieved in the XML document as the following ISO Date and Time Format: <code>YYYY-MM-DD&quot;T&quot;HH24:MI:SS.FF3</code> (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: <code>YYYY-MM-DD&quot;T&quot;HH24:MI:SS</code>  ■ NLS_TIMESTAMP_FORMAT - Must be set to the following Oracle Date and Time Format: <code>YYYY-MM-DD&quot;T&quot;HH24:MI:SS.FF3</code>  ■ NLS_TIMESTAMP_TZ_FORMAT - Must be set to the following Oracle Date and Time Format: <code>YYYY-MM-DD&quot;T&quot;HH24:MI:SS.FF3</code></td>
</tr>
</tbody>
</table>

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 3–14), 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:

/* Database Setup
drop table specialchars;
create table specialchars ('id' number, name varchar2(255));
insert into specialchars values (1, '<Jones>');
commit;
*/

// C#

using System;
using System.Data;
using System.Xml;
using Oracle.DataAccess.Client;

class QueryResultAsXMLSample
{
    static void Main()
    {
        //
    }
{  
  OracleConnection con = new OracleConnection();  
  con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";  
  con.Open();  
  // Create the command  
  OracleCommand cmd = new OracleCommand("", con);  
  // 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);  
  // Close and Dispose OracleConnection object  
  con.Close();  
  con.Dispose();  
}

The following XML document is generated for that table:

```xml
<?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 Oracle9i or Higher** When retrieving the query results as XML from Oracle9i 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 Oracle9i database, the SQL query in the following example can be used to get a result as an XML document from the specialchars table:

```sql
select 'some id', name from specialchars;
```

**See Also:**  "Special Characters in XML" on page 3-39

**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:

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

Features of Oracle Data Provider for .NET  3-43
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 3-16 lists the date and time format handling for different database releases.

<table>
<thead>
<tr>
<th>Table 3-16 Database Release Date and Time Differences When Saving Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Release</strong></td>
</tr>
<tr>
<td>Oracle8i release (8.1.7), Oracle9i release 1 (9.0.x)</td>
</tr>
<tr>
<td>Oracle9i release 2 (9.2.x) and Oracle Database 10g</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

#### 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:

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

The following XML document is generated:

```xml
<?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
<?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 3–14), 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.

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

The following XML document can be used to insert values (1, '7<Jones>') to the specialchars table:

```xml
<?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:

```sql
//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
<?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

Oracle9i 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.

```sql
// 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.

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

```sql
// 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
<?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, &lt;Jones&gt;) into the specialchars table using view2:

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

**Saving Changes to Oracle9i 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 Oracle9i 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, &lt;Jones&gt;) into the specialchars table.

```xml
<?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.

```csharp
using System;
using System.Data;
using System.Xml;
using Oracle.DataAccess.Client;

class InsertUsingXmlDocSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();

        con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
        con.Open();
        Console.WriteLine("Connected Successfully");

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

        // Set the XML command type to query.
        cmd.XmlCommandType = OracleXmlCommandType.Insert;
```
ODP.NET XML Support

// Set the XML document
cmd.CommandText = "<?xml version = '1.0'?>\n" + "<ROWSET>\n" + "<ROW>\n" +
"<id>2</id>\n" + "<NAME>&lt;Jones&gt;</NAME>\n" + "</ROW>\n" +
"</ROWSET>\n";
cmd.XmlSaveProperties.Table = "specialchars";
string[] ucols = new string[2];
ucols[0] = "id";
ucols[1] = "NAME";
cmd.XmlSaveProperties.UpdateColumnsList = ucols;
// Insert rows
int rows = cmd.ExecuteNonQuery();
Console.WriteLine("Number of rows inserted successfully : {0} ", rows);
// Close and Dispose OracleConnection object
con.Close();
con.Dispose();
}
}

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.

3-48 Oracle Data Provider for .NET Developer’s Guide


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 3–17 lists the maximums and minimums for Oracle `NUMBER` and .NET `Decimal`.

<table>
<thead>
<tr>
<th>Oracle NUMBER</th>
<th>.NET Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>9,9999999999999999999999999999999999 9999 e125</td>
</tr>
<tr>
<td></td>
<td>79,228,162,514,264,337,593,543,950,335</td>
</tr>
<tr>
<td>Minimum</td>
<td>-9,9999999999999999999999999999999999 9999 e125</td>
</tr>
<tr>
<td></td>
<td>-79,228,162,514,264,337,593,543,950,335</td>
</tr>
</tbody>
</table>

**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 3–18 lists the maximums and minimums for Oracle `Date` and .NET `DateTime`.

<table>
<thead>
<tr>
<th>Oracle Date</th>
<th>.NET DateTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>Dec 31, 9999 AD</td>
</tr>
<tr>
<td></td>
<td>Dec 31, 9999 AD 23:59:59.999999999</td>
</tr>
<tr>
<td>Minimum</td>
<td>Jan 1, 4712 BC</td>
</tr>
<tr>
<td></td>
<td>Jan 1, 0001 AD 00:00:00.000000000</td>
</tr>
</tbody>
</table>

**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 3–19 lists the maximums and minimums for Oracle TimeStamp and .NET DateTime.

<table>
<thead>
<tr>
<th>Oracle TimeStamp</th>
<th>.NET DateTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>Dec 31, 9999 AD 23:59:59.999999999</td>
</tr>
<tr>
<td>Minimum</td>
<td>Jan 1, 4712 BC 00:00:00.000000000</td>
</tr>
</tbody>
</table>

**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 3–20 lists the maximums and minimums for Oracle INTERVAL DAY TO SECOND and .NET DateTime.

<table>
<thead>
<tr>
<th>Oracle INTERVAL DAY TO SECOND</th>
<th>.NET TimeSpan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>+99999999979 23:59:59.999999999</td>
</tr>
<tr>
<td>Minimum</td>
<td>-99999999979 23:59:59.999999999</td>
</tr>
</tbody>
</table>

**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 4-95
- "SelectCommand" on page 4-97

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?
- Configuring PrimaryKey and Constraints Properties
- Updating Without PrimaryKey and Constraints Configuration

**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 3-52 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.

### 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 6-2

This section includes the following:
- Globalization Settings
- Globalization-Sensitive Operations

### Globalization Settings

An `OracleGlobalization` object can be used to represent the following:
- Client Globalization Settings
Globalization Support

- Session Globalization Settings
- Thread-Based Globalization Settings

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:

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

class ClientGlobalizationInfoSample
{
    static void Main()
    {
        OracleGlobalization ClientGlob = OracleGlobalization.GetClientInfo();

        Console.WriteLine("Client machine language: " + ClientGlob.Language);
        Console.WriteLine("Client character set: " + ClientGlob.ClientCharacterSet);
    }
}
```

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:

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

class SessionGlobalizationSample
{
    static void Main()
    {
        OracleConnection con = new OracleConnection();
con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
con.Open();

OracleGlobalization SessionGlob = con.GetSessionInfo();

// SetSessionInfo updates the Session with the new value
SessionGlob.DateFormat = "YYYY/MM/DD";
con.SetSessionInfo(SessionGlob);
Console.WriteLine("Date Format successfully changed for the session");

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

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:

// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ThreadBasedGlobalizationSample
{
    static void Main(string[] args)
    {
        // Set the thread’s DateFormat for the OracleDate constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "YYYY-MON-DD";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleDate from a string using the DateFormat specified.
        OracleDate date = new OracleDate("1999-DEC-01");
/ Set a different DateFormat for the thread
info.DateFormat = 'MM/DD/YYYY';
OracleGlobalization.SetThreadInfo(info);

// Print "12/01/1999"
Console.WriteLine(date.ToString());
}
}

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.

The thread 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 thread globalization settings.

For example, INTERVAL DAY TO SECOND data are not affected by thread settings since no format is applicable for this type. However, the DateFormat and NumericCharacters properties can impact the string representation of DATE and NUMBER types, respectively, when they are retrieved as strings into the DataSet through Safe Type Mapping.
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.

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 3-58

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.
This chapter describes the following Oracle Data Provider for .NET classes.

- OracleCommand Class
- OracleCommandBuilder Class
- OracleConnection Class
- OracleDataAdapter Class
- OracleDataReader Class
- OracleError Class
- OracleErrorCollection Class
- OracleException Class
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventHandler Delegate
- OracleParameter Class
- OracleParameterCollection Class
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventHandler Delegate
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventHandler Delegate
- OracleTransaction Class
- OracleCollectionType Enumeration
- OracleDbType Enumeration
- OracleParameterStatus Enumeration
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#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleCommandSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

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

        // Create the OracleCommand
        OracleCommand cmd = new OracleCommand(cmdQuery);
OracleCommand Class

```csharp
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));
}

// Clean up
reader.Dispose();
cmd.Dispose();
con.Dispose();
}
```

Requirements

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleCommand Constructors</td>
<td>Instantiates a new instance of OracleCommand class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleCommand Static Methods
OracleCommand static methods are listed in Table 4–2.

Table 4–2 OracleCommand Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

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

Table 4–3 OracleCommand Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRowid</td>
<td>Adds the ROWID as part of the select list</td>
</tr>
<tr>
<td>AddToStatementCache</td>
<td>Causes executed statements to be cached, when the property is set to True and statement caching is enabled</td>
</tr>
<tr>
<td>ArrayBindCount</td>
<td>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</td>
</tr>
<tr>
<td>BindByName</td>
<td>Specifies the binding method in the collection</td>
</tr>
<tr>
<td>CommandText</td>
<td>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</td>
</tr>
<tr>
<td>CommandTimeout</td>
<td>Specifies the number of seconds the command is allowed to execute before terminating the execution with an exception</td>
</tr>
<tr>
<td>CommandType</td>
<td>Specifies the command type that indicates how the CommandText property is to be interpreted</td>
</tr>
<tr>
<td>Connection</td>
<td>Specifies the OracleConnection object that is used to identify the connection to execute a command</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>FetchSize</td>
<td>Specifies the size of OracleDataReader's internal cache to store result set data</td>
</tr>
<tr>
<td>InitialLOBFetchSize</td>
<td>Specifies the amount that the OracleDataReader initially fetches for LOB columns</td>
</tr>
<tr>
<td>InitialLONGFetchSize</td>
<td>Specifies the amount that the OracleDataReader initially fetches for LONG and LONG RAW columns</td>
</tr>
</tbody>
</table>
OracleCommand Public Methods

OracleCommand public methods are listed in Table 4-4.

Table 4-4  OracleCommand Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel</td>
<td>Attempts to cancels a command that is currently executing on a particular connection</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of OracleCommand object</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>CreateParameter</td>
<td>Creates a new instance of OracleParameter class</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>ExecuteNonQuery</td>
<td>Executes a SQL statement or a command using the XmlCommandType and CommandText properties and returns the number of rows affected</td>
</tr>
<tr>
<td>ExecuteReader</td>
<td>Executes a command (Overloaded)</td>
</tr>
<tr>
<td>ExecuteScalar</td>
<td>Returns the first column of the first row in the result set returned by the query</td>
</tr>
<tr>
<td>ExecuteStream</td>
<td>Executes a command using the XmlCommandType and CommandText properties and returns the results in a new Stream object</td>
</tr>
<tr>
<td>ExecuteToStream</td>
<td>Executes a command using the XmlCommandType and CommandText properties and appends the results as an XML document to the existing Stream</td>
</tr>
<tr>
<td>ExecuteXmlReader</td>
<td>Executes a command using the XmlCommandType and CommandText properties and returns the result as an XML document in a .NET XmlTextReader object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Gettype</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Prepare</td>
<td>This method is a no-op</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- OracleCommand Class
- OracleCommand Members

OracleCommand()

This constructor instantiates a new instance of OracleCommand class.

Declarations

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

Remarks

Default constructor.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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.

Declarations

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

Parameters

- `cmdText`
The SQL command or stored procedure to be executed.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

```csharp
// 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" on page 1-1
- OracleCommand Class
- OracleCommand Members
OracleCommand Static Methods

OracleCommand static methods are listed in Table 4-5.

Table 4–5  OracleCommand Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommand Class
- OracleCommand Members
OracleCommand Properties

OracleCommand properties are listed in Table 4-6.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRowid</td>
<td>Adds the ROWID as part of the select list</td>
</tr>
<tr>
<td>AddToStatementCache</td>
<td>Causes executed statements to be cached, when the property is set to True</td>
</tr>
<tr>
<td></td>
<td>and statement caching is enabled</td>
</tr>
<tr>
<td>ArrayBindCount</td>
<td>Specifies if the array binding feature is to be used and also specifies</td>
</tr>
<tr>
<td></td>
<td>the maximum number of array elements to be bound in the Value property</td>
</tr>
<tr>
<td>BindByName</td>
<td>Specifies the binding method in the collection</td>
</tr>
<tr>
<td>CommandText</td>
<td>Specifies the SQL statement or stored procedure to run against the Oracle</td>
</tr>
<tr>
<td></td>
<td>database or the XML data used to store changes to the Oracle database</td>
</tr>
<tr>
<td>CommandType</td>
<td>Specifies the command type that indicates how the CommandText property</td>
</tr>
<tr>
<td></td>
<td>is to be interpreted</td>
</tr>
<tr>
<td>Connection</td>
<td>Specifies the OracleConnection object that is used to identify the</td>
</tr>
<tr>
<td></td>
<td>connection to execute a command</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>FetchSize</td>
<td>Specifies the size of OracleDataReader’s internal cache to store result set</td>
</tr>
<tr>
<td></td>
<td>data</td>
</tr>
<tr>
<td>InitialLOBFetchSize</td>
<td>Specifies the amount that the OracleDataReader initially fetches for LOB</td>
</tr>
<tr>
<td></td>
<td>columns</td>
</tr>
<tr>
<td>InitialLONGFetchSize</td>
<td>Specifies the amount that the OracleDataReader initially fetches for LONG</td>
</tr>
<tr>
<td></td>
<td>and LONG RAW columns</td>
</tr>
<tr>
<td>Parameters</td>
<td>Specifies the parameters for the SQL statement or stored procedure</td>
</tr>
<tr>
<td>RowSize</td>
<td>Specifies the amount of memory needed by the OracleDataReader internal</td>
</tr>
<tr>
<td></td>
<td>cache to store one row of data</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Transaction</td>
<td>Specifies the OracleTransaction object in which the OracleCommand executes</td>
</tr>
<tr>
<td>UpdatedRowSource</td>
<td>Specifies how query command results are applied to the row being updated</td>
</tr>
<tr>
<td>XmlCommandType</td>
<td>Specifies the type of XML operation on the OracleCommand</td>
</tr>
<tr>
<td>XmlQueryProperties</td>
<td>Specifies the properties that are used when an XML document is created</td>
</tr>
<tr>
<td></td>
<td>from the result set of a SQL query statement</td>
</tr>
<tr>
<td>XmlSaveProperties</td>
<td>Specifies the properties that are used when an XML document is used to</td>
</tr>
<tr>
<td></td>
<td>save changes to the database</td>
</tr>
</tbody>
</table>
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 ROWIDs of a table, the ROWID must explicitly be added to the select list without the use of this property.

---

AddToStatementCache

This property causes executed statements to be cached when the property is set to `True` and statement caching is enabled. If statement caching is disabled or if this property is set to `False`, the executed statement is not cached.

**Declaration**

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

**Return Value**

Returns `bool` value. A value of `true` indicates that statements are being added to the cache, `false` indicates otherwise.

**Property Value**

A `bool` value that indicates that the statements will be cached when they are executed, if statement caching is enabled.

**Remarks**

Default is `true`. 

---

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommand Class
- OracleCommand Members

---

"LOB Support" on page 3-32 for further information on how this property used with LOBs
AddToStatementCache is ignored if statement caching is disabled. Statement caching is enabled by setting the Statement Cache Size connection string attribute to a value greater than 0.

When statement caching is enabled, however, this property provides a way to selectively add statements to the cache.

**Example**

```csharp
// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class AddToStatementCacheSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle;" +
            "statement cache size=10";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand("select * from emp", con);

        // The execution of "select * from emp" will be added to the statement cache
        // because statement cache size is greater than 0 and OracleCommand's
        // AddToStatementCache is true by default.
        OracleDataReader readerEmp = cmd.ExecuteReader();

        // Do not add "select * from dept" to the statement cache
        cmd.CommandText = "select * from dept";
        cmd.AddToStatementCache = false;

        // The execution of "select * from dept" will not be added to the
        // statement cache because AddToStatementCache is set to false.
        OracleDataReader readerDept = cmd.ExecuteReader();

        // Clean up
        con.Dispose();
    }
}
```

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommand Class
- OracleCommand Members
- "Statement Caching" on page 3-21
- ConnectionString on page 4-62

**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" on page 1-1
- OracleCommand Class
- OracleCommand Members
- "Array Binding" on page 3-18
- "Value" on page 4-62

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 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" on page 1-1
- `OracleCommand Class`
- `OracleCommand Members`
- "Array Binding" on page 3-18
- "Value" on page 4-62

**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**

```csharp
// 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.
CommandTimeout

This property specifies the number of seconds that the command is allowed to execute before terminating with an exception.

**Declaration**

```csharp
public int CommandTimeout {get; set;}
```

**Property Value**

`int`

**Implements**

`IDbCommand.CommandTimeout`

**Exception**

`ArgumentException` - The specified value is less than 0.

**Remarks**

Default is 0 seconds, which enforces no time limit.

When the specified timeout value expires before a command execution finishes, the command attempts to cancel. If cancellation is successful, an exception is thrown with the message of **ORA-01013: user requested cancel of current operation** error. If the command executed in time without any errors, no exceptions are thrown.

In a situation where multiple `OracleCommand` objects use the same connection, the timeout expiration on one of the `OracleCommand` objects may terminate any of the executions on the single connection. To make the timeout expiration of a `OracleCommand` cancel only its own command execution, simply use one `OracleCommand` for each connection if that `OracleCommand` sets the `CommandTimeout` property to a value greater than 0.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-1
- `OracleCommand Class`
- `OracleCommand Members`

CommandType

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

**Declaration**

```
// C#
public System.Data.CommandType CommandType {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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- OracleCommand Class
- OracleCommand Members
- OracleDataReader "FetchSize" on page 4-117

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" on page 1-1
- OracleCommand Class
- OracleCommand Members
- "Obtaining LOB Data" on page 3-27

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" on page 1-1
- OracleCommand Class
- OracleCommand Members
- "Obtaining LONG and LONG RAW Data" on page 3-27 for further information

### Parameters

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

**Declaration**

```csharp
// 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" on page 1-1
- 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**

```csharp
// 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" on page 1-1
- OracleCommand Class
- OracleCommand Members
- OracleDataReader "FetchSize" on page 4-16

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" on page 1-1
- 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" on page 1-1
- 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.
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" on page 1-1
- 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" on page 1-1
- OracleCommand Class
- OracleCommand Members
OracleCommand Public Methods

OracleCommand public methods are listed in Table 4–7.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel</td>
<td>Attempts to cancels a command that is currently executing on a particular connection</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of OracleCommand object</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>CreateParameter</td>
<td>Creates a new instance of OracleParameter class</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>ExecuteNonQuery</td>
<td>Executes a SQL statement or a command using the XmlCommandType and CommandText properties and returns the number of rows affected</td>
</tr>
<tr>
<td>ExecuteReader</td>
<td>Executes a command (Overloaded)</td>
</tr>
<tr>
<td>ExecuteScalar</td>
<td>Returns the first column of the first row in the result set returned by the query</td>
</tr>
<tr>
<td>ExecuteStream</td>
<td>Executes a command using the XmlCommandType and CommandText properties and returns the results in a new Stream object</td>
</tr>
<tr>
<td>ExecuteToStream</td>
<td>Executes a command using the XmlCommandType and CommandText properties and appends the results as an XML document to the existing Stream</td>
</tr>
<tr>
<td>ExecuteXmlReader</td>
<td>Executes a command using the XmlCommandType and CommandText properties and returns the result as an XML document in a .NET XmlTextReader object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Prepare</td>
<td>This method is a no-op</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommand Class
- OracleCommand Members

Cancel

This method attempts to cancel a command that is currently executing on a particular connection.
Declaration

// C#
public void Cancel();

Implements
IDbCommand.Cancel

Remarks
Invoking the Cancel method does not guarantee that the command executing at the time is actually cancelled. If the cancellation of the command succeeds, an exception is thrown. If the cancellation is not successful, no exception is thrown. If there is no command being executed at the time of the Cancel invocation, Cancel does nothing.

When multiple OracleCommand objects share the same connection, only one command can be executed on that connection at any one time. When it is invoked, the Cancel method attempts to cancel the statement currently running on the connection that the OracleCommand object is using to execute the command. However, when multiple OracleCommand objects execute statements on the same connection simultaneously, issuing a Cancel method invocation may cancel any of the issued commands. This is because the currently running statement may have completed and the next command started executing by the time the cancel request reaches the server.

There are several ways to eliminate this non-deterministic situation as follows:

- If Cancel is used on that command, the application can create just one OracleCommand object for each connection or the command executions between OracleCommand objects that use the same connection can be synchronized.
- If Cancel is not used, there is no need to synchronize or limit one OracleCommand object for each connection.

Since the termination on the currently running execution is non-deterministic, it is recommended that any non-atomic SQL or PL/SQL execution be started within a transaction. When the command execution successfully terminates with an exception of ORA-01013: user requested cancel of current operation error, the transaction can be rolled back for data integrity. Examples of non-atomic execution are collections of DML command executions that are executed one-by-one and multiple DML commands that are part of a PL/SQL stored procedure or function.

Example

// C#

// This example shows how command executions can be cancelled in a deterministic way even if multiple commands are executed on a single connection.
// This is accomplished by synchronizing threads through events.
// Since the Cancel method terminates the currently running operation on the connection, threads must be serialized if multiple threads are using the same connection to execute server round-trip incurring operations.
// Furthermore, the example shows how the execution and cancel threads should be synchronized so that nth iteration of the command execution does not inappropriately cancel the (n+1)th command executed by the same thread.

using System;
using System.Data;
using Oracle.DataAccess.Client;
using System.Threading;
class CancelSample
{
    private OracleCommand cmd;
    Thread t1, t2;
    // threads signal following events when assigned operations are completed

    private AutoResetEvent ExecuteEvent = new AutoResetEvent(false);
    private AutoResetEvent CancelEvent = new AutoResetEvent(false);
    private AutoResetEvent FinishedEvent = new AutoResetEvent(false);
    AutoResetEvent[] ExecuteAndCancel = new AutoResetEvent[2];

    // Default constructor
    CancelSample()
    {
        cmd = new OracleCommand("select * from all_objects",
                                new OracleConnection("user id=scott;password=tiger;data source=oracle"));
        ExecuteAndCancel[0] = ExecuteEvent;
        ExecuteAndCancel[1] = CancelEvent;
    }

    // Constructor that takes a particular command and connection
    CancelSample(string command, OracleConnection con)
    {
        cmd = new OracleCommand(command, con);
        ExecuteAndCancel[0] = ExecuteEvent;
        ExecuteAndCancel[1] = CancelEvent;
    }

    // Execution of the command
    public void Execute()
    {
        OracleDataReader reader = null;
        try
        {
            Console.WriteLine("Execute.");
            reader = cmd.ExecuteReader();
            Console.WriteLine("Execute Done.");
            reader.Close();
        }
        catch(Exception e)
        {
            Console.WriteLine("The command has been cancelled.", e.Message);
        }
        ExecuteEvent.Set();
    }

    // Canceling of the command
    public void Cancel()
    {
        try
        {
            // cancel query if it takes longer than 100 ms to finish execution
            System.Threading.Thread.Sleep(100);
            Console.WriteLine("Cancel.");
            cmd.Cancel();
        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }
    }
}
```csharp
ExecuteWithinLimitedTime()
{    for (int i = 0; i < 5; i++)    {        Monitor.Enter(typeof(CancelSample));        try        {            Console.WriteLine("Executing " + this.cmd.CommandText);            ExecuteEvent.Reset();            CancelEvent.Reset();            t1 = new Thread(new ThreadStart(this.Execute));            t2 = new Thread(new ThreadStart(this.Cancel));            t1.Start();            t2.Start();        } finally        {            WaitHandle.WaitAll(ExecuteAndCancel);            Monitor.Exit(typeof(CancelSample));        }    }    FinishedEvent.Set();}

[Multithreaded]
static void Main()
{
    try
    {
        AutoResetEvent[] ExecutionCompleteEvents = new AutoResetEvent[3];

        // Create the connection that is to be used by three commands
        OracleConnection con = new OracleConnection("user id=scott;" +
            "password=tiger;data source=oracle");
        con.Open();

        // Create instances of CancelSample class
        CancelSample test1 = new CancelSample("select * from all_objects", con);
        CancelSample test2 = new CancelSample("select * from all_objects, emp", con);
        CancelSample test3 = new CancelSample("select * from all_objects, dept", con);

        // Create threads for each CancelSample object instance
        Thread t1 = new Thread(new ThreadStart(test1.ExecuteWithinLimitedTime));
        Thread t2 = new Thread(new ThreadStart(test2.ExecuteWithinLimitedTime));
        Thread t3 = new Thread(new ThreadStart(test3.ExecuteWithinLimitedTime));

        // Obtain a handle to an event from each object
        ExecutionCompleteEvents[0] = test1.FinishedEvent;
        ExecutionCompleteEvents[1] = test2.FinishedEvent;

        // Start all threads to execute three commands using a single connection
    }
```
t1.Start();
t2.Start();
t3.Start();

// Wait for all three commands to finish executing/canceling before
// closing the connection
WaitHandle.WaitAll(ExecutionCompleteEvents);
con.Close();
}
catch (Exception e)
{
    Console.WriteLine(e.ToString());
}
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommand Class
- OracleCommand Members
- http://msdn.microsoft.com/library for detailed information about this Microsoft .NET Framework 1.1 feature

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.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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**

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

class ExecuteNonQuerySample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand(
            "select sal from emp where empno=7934", con);
        object sal = cmd.ExecuteScalar();
        Console.WriteLine("Employee sal before update: " + sal);

        cmd.CommandText = "update emp set sal = sal + .01 where empno=7934";
        // Auto-commit changes
        int rowsUpdated = cmd.ExecuteNonQuery();
        if (rowsUpdated > 0)
        {
            cmd.CommandText = "select sal from emp where empno=7934";
            sal = cmd.ExecuteScalar();
            Console.WriteLine("Employee sal after update: " + sal);
        }

        // Clean up
        cmd.Dispose();
        con.Dispose();
    }
}
```

**Requirements**

For XML support, this method requires Oracle9i XML Developer's Kits (Oracle XDK) or higher, to be installed in the database. Oracle XDK can be downloaded from Oracle Technology Network (OTN).
ExecuteReader

Overload List:
ExecuteReader executes a command specified in the CommandText.

- **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" on page 1-1
- OracleCommand Class
- OracleCommand Members

ExecuteReader()

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

Declaration

```csharp
// 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 3-27.

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**

```csharp
// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class ExecuteReaderSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand("select ename from emp", con);

        OracleDataReader reader = cmd.ExecuteReader();

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

        // Clean up
        reader.Dispose();
        cmd.Dispose();
        con.Dispose();
    }
}
```

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommand Class
- OracleCommand Members
- "OracleRefCursor Class" on page 8-109

**ExecuteReader(CommandBehavior)**

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

**Declaration**

```csharp
// 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 8-109.

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 3-27.

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" on page 1-1
- OracleCommand Class
- OracleCommand Members
- "OracleRefCursor Class" on page 8-109

**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**

```csharp
// 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

```csharp
// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class ExecuteScalarSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand("select count(*) from emp", con);
        object count = cmd.ExecuteScalar();

        Console.WriteLine("There are {0} rows in table emp", count);

        // Clean up
        cmd.Dispose();
        con.Dispose();
    }
}
```

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

```csharp
// 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 `ExecuteStream` 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" on page 1-1
- `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" on page 1-1
- 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`
  Ignored.
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" on page 1-1
- OracleCommand Class
- OracleCommand Members
- Oracle XML DB Developer’s Guide
- http://otn.oracle.com/
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 following example performs an update on the EMP table. It uses the OracleCommandBuilder object to create the UpdateCommand for the OracleDataAdapter object when OracleDataAdapter.Update() is called.

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleCommandBuilderSample
{
    static void Main()


```csharp
{
    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    string cmdstr = "SELECT empno, sal from emp";

    // Create the adapter with the selectCommand txt and the
    // connection string
    OracleDataAdapter adapter = new OracleDataAdapter(cmdstr, constr);

    // 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 EMP
    DataSet dataset = new DataSet();
    adapter.Fill(dataset, "EMP");

    // Get the EMP table from the dataset
    DataTable table = dataset.Tables["EMP"];

    // Indicate DataColumn EMPNO is unique
    // This is required by the OracleCommandBuilder to update the EMP table
    table.Columns["EMPNO"].Unique = true;

    // Get the first row from the EMP table
    DataRow row = table.Rows[0];

    // Update the salary
    double sal = double.Parse(row["SAL"].ToString());
    row["SAL"] = sal + .01;

    // Now update the EMP using the adapter
    // The OracleCommandBuilder will create the UpdateCommand for the
    // adapter to update the EMP table
    adapter.Update(dataset, "EMP");

    Console.WriteLine("Row updated successfully");
}
```

**Requirements**

**Namespace:** Oracle.DataAccess.Client

**Assembly:** Oracle.DataAccess.dll

**See Also:**

- "Oracle.DataAccess.Client Namespace” on page 1-1
- 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**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleCommandBuilder Constructors</td>
<td>Instantiates a new instance of OracleCommandBuilder class (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleCommandBuilder Static Methods**

OracleCommandBuilder static methods are listed in Table 4–9.

**Table 4–9 OracleCommandBuilder Static Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeriveParameters</td>
<td>Queries for the parameters of a stored procedure or function, represented by a specified OracleCommand, and populates the OracleParameterCollection of the command with the return values</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleCommandBuilder Properties**

OracleCommandBuilder properties are listed in Table 4–10.

**Table 4–10 OracleCommandBuilder Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>DataAdapter</td>
<td>Indicates the OracleDataAdapter for which the SQL statements are generated</td>
</tr>
<tr>
<td>CaseSensitive</td>
<td>Indicates whether or not double quotes are used around Oracle object names when generating SQL statements</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from Component</td>
</tr>
</tbody>
</table>

**OracleCommandBuilder Public Methods**

OracleCommandBuilder public methods are listed in Table 4–11.

**Table 4–11 OracleCommandBuilder Public Methods**

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetDeleteCommand</td>
<td>Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform deletions on the database</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>
OracleCommandBuilder Class

OracleCommandBuilder Events
OracleCommandBuilder events are listed in Table 4–12.

Table 4–12  OracleCommandBuilder Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from Component</td>
</tr>
</tbody>
</table>

OracleCommandBuilder Event Delegates
OracleCommandBuilder event delegates are listed in Table 4–13.

Table 4–13  OracleCommandBuilder Event Delegates

<table>
<thead>
<tr>
<th>Event Delegate Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHandler</td>
<td>Inherited from Component</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
  - OracleCommandBuilder Class
  - OracleCommandBuilder Members

**OracleCommandBuilder()**
This constructor creates an instance of the OracleCommandBuilder class.

**Declaration**

```csharp
// C#
public OracleCommandBuilder();
```

**Remarks**
Default constructor.

  **See Also:**
  - "Oracle.DataAccess.Client Namespace" on page 1-1
  - 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**

```csharp
// C#
public OracleCommandBuilder(OracleDataAdapter da);
```

**Parameters**
- **da**
  The OracleDataAdapter object provided.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- `OracleCommandBuilder Class`
- `OracleCommandBuilder Members`
OracleCommandBuilder Static Methods

OracleCommandBuilder static methods are listed in Table 4-14.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeriveParameters</td>
<td>Queries for the parameters of a stored procedure or function, represented by a specified OracleCommand, and populates the OracleParameterCollection of the command with the return values</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommandBuilder Class
- OracleCommandBuilder Members

DeriveParameters

This method queries for the parameters of a stored procedure or function, represented by a specified OracleCommand, and populates the OracleParameterCollection of the command with the return values.

Declaration

```csharp
// C#
public static void DeriveParameters(OracleCommand command);
```

Parameters

- `command`
  The command that represents the stored procedure or function for which parameters are to be derived.

Exceptions

- `InvalidOperationException`: The CommandText is not a valid stored procedure or function name, the CommandType is not CommandType.StoredProcedure, or the Connection.State is not ConnectionState.Open.

Remarks

DeriveParameters can only be used for stored procedures or functions, not for anonymous PL/SQL blocks.

Invoking DeriveParameters deletes all existing parameters in the parameter collection of the command.

DeriveParameters incurs a server round-trip and should only be used during design time. To avoid unnecessary server round-trips in a production environment, the DeriveParameters method itself should be replaced with the explicit parameter settings that were returned by the DeriveParameters method at design time.

DeriveParameters can only preserve the case of the stored procedure or function name if it is encapsulated by double-quotes. For example, if the stored procedure in
the database is named GetEmployees with mixed-case, the CommandText property on the OracleCommand object must be set appropriately as in the following example:

```csharp
cmd.CommandText = "\"GetEmployees\"";
```

Stored procedures and functions in a package must be provided in the following format:

```csharp
<package name>.<procedure or function name>
```

For example, to obtain parameters for a stored procedure named GetEmployees (mixed-case) in a package named EmpProcedures (mixed-case), the name provided to the OracleCommand is:

```csharp
'\"EmpProcedures\".'\"GetEmployees\"'
```

DeriveParameters cannot be used for object type methods.

The derived parameters contain all the metadata information that is needed for the stored procedure to execute properly. The application must provide the value of the parameters before execution, if required.

The output values of derived parameters return as .NET Types by default. To obtain output parameters as provider types, the OracleDbType property of the parameter must be set explicitly by the application to override this default behavior. One quick way to do this is to set the OracleDbType to itself for all output parameters that should be returned as provider types.

The BindByName property of the supplied OracleCommand is left as is, but the application can change its value.

If the specified stored procedure or function is overloaded, the first overload is used to populate the parameters collection.

**Example**

```csharp
// Database Setup
/*
CREATE OR REPLACE PROCEDURE MyOracleStoredProc (arg_in IN VARCHAR2,
    arg_out OUT VARCHAR2) IS
    BEGIN
    arg_out := arg_in;
END;
*/

// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class DeriveParametersSample
{
    static void Main()
    {
        // Create the PL/SQL Stored Procedure MyOracleStoredProc as indicated above

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();
    }
```
// Create an OracleCommand
OracleCommand cmd = new OracleCommand("MyOracleStoredProc", con);
cmd.CommandType = CommandType.StoredProcedure;

// Derive Parameters
OracleCommandBuilder.DeriveParameters(cmd);
Console.WriteLine("Parameters Derived");

// Prints "Number of Parameters for MyOracleStoredProc = 2"
Console.WriteLine("Number of Parameters for MyOracleStoredProc = {0}",
    cmd.Parameters.Count);

// Modify the OracleParameter Properties, if applicable
// Set the Value for the IN parameter.
// The Size can also be set, this sample uses the default size
cmd.Parameters[0].Value = "MyText";

// Prints "cmd.Parameters[1].Size = 4000"
// The default size for OUT VARCHAR2 is 4000
Console.WriteLine("cmd.Parameters[1].Size = " + cmd.Parameters[1].Size);

// Set the Size for the OUT parameter
cmd.Parameters[1].Size = 6;

// Execute the command
cmd.ExecuteNonQuery();

// Prints "cmd.Parameters[1].Value = MyText"
Console.WriteLine("cmd.Parameters[1].Value = " + cmd.Parameters[1].Value);

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommandBuilder Class
- OracleCommandBuilder Members
- OracleCommand Class
- OracleParameter Class
- OracleParameterCollection Class
- http://msdn.microsoft.com/library for detailed information about this Microsoft .NET Framework 1.1 feature
OracleCommandBuilder Class

OracleCommandBuilder Properties

OracleCommandBuilder properties are listed in Table 4–15.

Table 4–15    OracleCommandBuilder Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>DataAdapter</td>
<td>Indicates the OracleDataAdapter for which the SQL statements are generated</td>
</tr>
<tr>
<td>CaseSensitive</td>
<td>Indicates whether or not double quotes are used around Oracle object names when generating SQL statements</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from Component</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- OracleCommandBuilder Class
- OracleCommandBuilder Members
OracleCommandBuilder Public Methods

OracleCommandBuilder public methods are listed in Table 4–16.

Table 4–16  OracleCommandBuilder Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetDeleteCommand</td>
<td>Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform deletions on the database</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetInsertCommand</td>
<td>Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform insertions on the database</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetUpdateCommand</td>
<td>Gets the automatically generated OracleCommand object that has the SQL statement (CommandText) perform updates on the database</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>RefreshSchema</td>
<td>Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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.
InvalidOperation Exception - 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" on page 1-1
- OracleCommandBuilder Class
- OracleCommandBuilder Members

RefreshSchema
This method refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements.

Declaration
```csharp
// C#
public void RefreshSchema();
```

Remarks
An application should call RefreshSchema whenever the SelectCommand value of the OracleDataAdapter changes.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommandBuilder Class
- OracleCommandBuilder Members
OracleCommandBuilder Events

OracleCommandBuilder events are listed in Table 4–17.

Table 4–17  OracleCommandBuilder Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from Component</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleCommandBuilder Class
- OracleCommandBuilder Members
OracleCommandBuilder Event Delegates

OracleCommandBuilder event delegates are listed in Table 4–18.

Table 4–18 OracleCommandBuilder Event Delegates

<table>
<thead>
<tr>
<th>Event Delegate Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHandler</td>
<td>Inherited from Component</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleConnectionSample
{
  static void Main()
  {
    // Connect
    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    // Execute a SQL SELECT
    OracleCommand cmd = con.CreateCommand();
    cmd.CommandText = "select * from emp";
    OracleDataReader reader = cmd.ExecuteReader();

    // Print all employee numbers
    while (reader.Read())
      Console.WriteLine(reader.GetInt32(0));

    // Clean up
    reader.Dispose();
    cmd.Dispose();
    con.Dispose();
  }
}

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleConnection Constructors</td>
<td>Instantiates a new instance of the OracleConnection class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleConnection Static Methods
OracleConnection static methods are listed in Table 4–20.

Table 4–20 OracleConnection Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleConnection Properties
OracleConnection properties are listed in Table 4–21.

Table 4–21 OracleConnection Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionString</td>
<td>Specifies connection information used to connect to an Oracle database</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Specifies the maximum amount of time that the Open() method can take to obtain a pooled connection before terminating the request</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>DataSource</td>
<td>Specifies the Oracle Net Service Name (also known as TNS alias) that identifies an Oracle database instance</td>
</tr>
<tr>
<td>ServerVersion</td>
<td>Specifies the version number of the Oracle database to which the OracleConnection has established a connection</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>State</td>
<td>Specifies the current state of the connection</td>
</tr>
</tbody>
</table>

OracleConnection Public Methods
OracleConnection public methods are listed in Table 4–22.

Table 4–22 OracleConnection Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginTransaction</td>
<td>Begins a local transaction (Overloaded)</td>
</tr>
<tr>
<td>ChangeDatabase</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleConnection object</td>
</tr>
</tbody>
</table>
Table 4–22 (Cont.) OracleConnection Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Closes the database connection</td>
</tr>
<tr>
<td>CreateCommand</td>
<td>Creates and returns an OracleCommand object associated with the OracleConnection object</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>EnlistDistributedTransaction</td>
<td>Enables applications to explicitly enlist in a specified distributed transaction</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetSessionInfo</td>
<td>Returns or refreshes the property values of the OracleGlobalization object that represents the globalization settings of the session (Overloaded)</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Open</td>
<td>Opens a database connection with the property settings specified by the ConnectionString</td>
</tr>
<tr>
<td>OpenWithNewPassword</td>
<td>Opens a new connection with the new password</td>
</tr>
<tr>
<td>PurgeStatementCache</td>
<td>Flushes the Statement Cache by closing all open cursors on the server, when Statement Caching is enabled</td>
</tr>
<tr>
<td>SetSessionInfo</td>
<td>Alters the session’s globalization settings with the property values provided by the OracleGlobalization object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

OracleConnection Events

OracleConnection events are listed in Table 4–23.

Table 4–23 OracleConnection Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Failover</td>
<td>An event that is triggered when an Oracle failover occurs</td>
</tr>
<tr>
<td>InfoMessage</td>
<td>An event that is triggered for any message or warning sent by the database</td>
</tr>
<tr>
<td>StateChange</td>
<td>An event that is triggered when the connection state changes</td>
</tr>
</tbody>
</table>

OracleConnection Event Delegates

OracleConnection event delegates are listed in Table 4–24.
Table 4–24  OracleConnection Event Delegates

<table>
<thead>
<tr>
<th>Event Delegate Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleFailoverEventHandler</td>
<td>An event delegate that handles the Failover event</td>
</tr>
<tr>
<td>OracleInfoMessageEventHandler</td>
<td>An event delegate that handles the InfoMessage event</td>
</tr>
<tr>
<td>StateChangeEventHandler</td>
<td>An event delegate that handles the StateChange event</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
  - OracleConnection Class
  - OracleConnection Members

OracleConnection()

This constructor instantiates a new instance of the OracleConnection class using default property values.

**Declaration**

```csharp
// 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" on page 1-1
  - OracleConnection Class
  - OracleConnection Members

OracleConnection(String)

This constructor instantiates a new instance of the OracleConnection class with the provided connection string.

**Declaration**

```csharp
// 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" on page 1-1
- `OracleConnection Class`
- `OracleConnection Members`
OracleConnection Static Methods

OracleConnection static methods are listed in Table 4–25.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members
OracleConnection Properties

OracleConnection properties are listed in Table 4–26

Table 4–26 OracleConnection Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionString</td>
<td>Specifies connection information used to connect to an Oracle database</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Specifies the maximum amount of time that the Open() method can take to obtain a pooled connection before terminating the request</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>DataSource</td>
<td>Specifies the Oracle Net Service Name (also known as TNS alias) that identifies an Oracle database instance</td>
</tr>
<tr>
<td>ServerVersion</td>
<td>Specifies the version number of the Oracle database to which the OracleConnection has established a connection</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>State</td>
<td>Specifies the current state of the connection</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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:
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:

```
'Struct 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 Ids** that are not doubled-quoted are converted to all upper case when connecting.

Single quotes are not supported.

For a complete example, see "**Example**" on page 4-66.

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>
<thead>
<tr>
<th>Connection String Attribute</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Lifetime</td>
<td>0</td>
<td>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.</td>
</tr>
</tbody>
</table>
OracleConnection Properties

**Table 4–27  (Cont.) Supported Connection String Attributes**

<table>
<thead>
<tr>
<th>Connection String Attribute</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection Timeout</strong></td>
<td>15</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Data Source</strong></td>
<td>empty string</td>
<td>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.</td>
</tr>
<tr>
<td><strong>DBA Privilege</strong></td>
<td>empty string</td>
<td>Administrative privileges SYSDBA or SYSOPER. This connection string attribute only accepts SYSDBA or SYSOPER as the attribute value. It is case insensitive.</td>
</tr>
<tr>
<td><strong>Decr Pool Size</strong></td>
<td>1</td>
<td>Number of connections that are closed when an excessive amount of established connections are unused. 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.</td>
</tr>
<tr>
<td><strong>Enlist</strong></td>
<td>True</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Incr Pool Size</strong></td>
<td>5</td>
<td>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.</td>
</tr>
</tbody>
</table>
Max Pool Size 100

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.

Min Pool Size 1

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

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 UserId attribute. ODP.NET attempts to establish a proxy connection if either the Proxy UserId or the Proxy Password attribute is set to a non-empty string.

For the proxy user to connect to an Oracle database using operating system authentication, the ProxyUserId must be set to "/". The Proxy Password is ignored in this case. The UserId cannot be set to "/" when establishing proxy connections. The case of this attribute value is preserved if it is surrounded by double quotes.
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.

```csharp
/* Database Setup
grant connect , resource to 'MYSCHEMA' identified by MYPWD;
grant connect , resource to 'MySchema' identified by MYPWD;
*/

// C#
using System;
using Oracle.DataAccess.Client;

class ConnectionStringSample
{
    static void Main()
    {
        string constr1 = "User Id=myschema;Password=mypwd;Data Source=oracle";
        string constr2 = "User Id=MySchema;Password=MyPwd;Data Source=oracle";
        string constr3 = "User Id=\'MYSCHEMA\';Password=MYPWD;Data Source=oracle";
        string constr4 = "User Id=\'MySchema\';Password=MYPWD;Data Source=oracle";
```
string constr5 = "User Id=myschema;Password=mypwd;Data Source=oracle; ";
string constr6 =
    "User Id=myschema;Password=mypwd;Data Source=oracle;pooling=true";

// Connect as 'MYSCHEMA/MYPWD'
// NOTE: the password is case insensitive
// A new connection and a new connection pool X is created
OracleConnection con1 = new OracleConnection(constr1);
con1.Open();

// Place connection back into connection pool X
con1.Dispose();

// Connect as 'MYSCHEMA/MYPWD'
// The connection from pool X is obtained. No new connection created.
OracleConnection con2 = new OracleConnection(constr2);
con2.Open();

// Place connection back into connection pool X
con2.Dispose();

// Connect as 'MYSCHEMA/MYPWD'
// The connection from pool X is obtained. No new connection created.
OracleConnection con3 = new OracleConnection(constr3);
con3.Open();

// Place connection back into connection pool X
con3.Dispose();

// Connect as 'MySchema/MYPWD'
// A new connection and connection pool Y is created
OracleConnection con4 = new OracleConnection(constr4);
con4.Open();

// Place connection back into connection pool Y
con4.Dispose();

// Connect as 'MYSCHEMA/MYPWD'
// The connection from pool X is obtained
// Extra spaces or semi-colons in the connection string do not force
// new pools to be created
OracleConnection con5 = new OracleConnection(constr5);
con5.Open();

// Place connection back into connection pool X
con5.Dispose();

// Connect as 'MYSCHEMA/MYPWD'
// A connection is obtained from Connection Pool X.
// It's important to note that different connection strings do
// not necessarily mean that ODP.NET will create different
// connection pools for them. In other words, ODP.NET does not
// use exact string matching algorithm to determine whether
// a new connection pool needs to be created or not.
// Instead, it creates connection pools based on the uniqueness
// of attribute values settings in the connection string.
OracleConnection con6 = new OracleConnection(constr6);
con6.Open();

// Place connection back into connection pool X
OracleConnection Properties

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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.
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" on page 1-1
- 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" on page 1-1
- OracleConnection Class
- OracleConnection Members
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members
OracleConnection Public Methods

OracleConnection public methods are listed in Table 4–28.

Table 4–28  OracleConnection Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginTransaction</td>
<td>Begins a local transaction (Overloaded)</td>
</tr>
<tr>
<td>ChangeDatabase</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleConnection object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the database connection</td>
</tr>
<tr>
<td>CreateCommand</td>
<td>Creates and returns an OracleCommand object associated with the OracleConnection object</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>EnlistDistributedTransaction</td>
<td>Enables applications to explicitly enlist in a specified distributed transaction</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetSessionInfo</td>
<td>Returns or refreshes the property values of the OracleGlobalization object that represents the globalization settings of the session (Overloaded)</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Open</td>
<td>Opens a database connection with the property settings specified by the ConnectionString</td>
</tr>
<tr>
<td>OpenWithNewPassword</td>
<td>Opens a new connection with the new password</td>
</tr>
<tr>
<td>PurgeStatementCache</td>
<td>Flushes the Statement Cache by closing all open cursors on the server, when Statement Caching is enabled</td>
</tr>
<tr>
<td>SetSessionInfo</td>
<td>Alters the session’s globalization settings with the property values provided by the OracleGlobalization object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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**

```csharp
// 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" on page 1-1
- `OracleConnection Class`
- `OracleConnection Members`

### BeginTransaction(IsolationLevel)

This method begins a local transaction with the specified isolation level.

**Declaration**

```csharp
// 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

```csharp
// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class BeginTransactionSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        // Start a transaction
        OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

        // Update EMP table
        cmd.CommandText = "update emp set sal = sal + 100";
        cmd.ExecuteNonQuery();

        // Rollback transaction
        txn.Rollback();
        Console.WriteLine("Transaction rolledback");

        // Clean up
        txn.Dispose();
        cmd.Dispose();
        con.Dispose();
    }
}
```

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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#
using System;
using Oracle.DataAccess.Client;

class CloneSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Need a proper casting for the return value when cloned
        OracleConnection clonedCon = (OracleConnection)con.Clone();

        // Cloned connection is always closed, regardless of its source,
        // But the connection string should be identical
        clonedCon.Open();

        // Close and Dispose OracleConnection object
        clonedCon.Dispose();
    }
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- `OracleConnection Class`
- `OracleConnection Members`

CreateCommand

This method creates and returns an `OracleCommand` object associated with the `OracleConnection` object.

Declaration

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

Return Value

The `OracleCommand` object.

Implements

`IDbConnection`

Example

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

class CreateCommandSample
{
    static void Main()
    {
        // Connect
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Execute a SQL SELECT
        OracleCommand cmd = con.CreateCommand();
        cmd.CommandText = "select * from emp";
        OracleDataReader reader = cmd.ExecuteReader();

        // Print all employee numbers
        while (reader.Read())
        {
            // ...
        }
    }
}
```
Console.WriteLine(reader.GetInt32(0));

// Clean up
reader.Dispose();
cmd.Dispose();
con.Dispose();
}
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members

EnlistDistributedTransaction
This method enables applications to explicitly enlist in a specific distributed transaction after a connection has been opened.

Declaration
// C#
public void EnlistDistributedTransaction(ITransaction transaction);

Parameters
- transaction
  An ITransaction interface

Exception
InvalidOperationException - The connection is part of a local transaction.

Remarks
EnlistDistributedTransaction enables objects to enlist in a specific transaction that is passed to the method. The ITransaction interface can be obtained by applying an (ITransaction) cast to the ContextUtil.Transaction property within the component that started the distributed transaction.

The connection must be open before calling this method or an InvalidOperationException is thrown.

If a connection is part of a local transaction that was started implicitly or explicitly while attempting to enlist in a distributed transaction, the local transaction is rolled back and an exception is thrown.

By default, distributed transactions roll back, unless the method-level AutoComplete declaration is set.

Invoking the commit on the ITransaction raises an exception.

Invoking the rollback on the ITransaction method and calling ContextUtil.SetComplete on the same distributed transaction raises an exception.

Example
Application:
// C#

// This is the class that will utilize the Enterprise Services component. This module needs to be built as an executable.

using System;
using System.EnterpriseServices;
using ConnectionSample_Comp;

class EnlistDistributedTransactionSample_App
{
    static void Main()
    {
        ESComponent comp = new ESComponent();
        comp.DoWork();
    }
}

Component:

// C#

// This module needs to be
// 1) built as a component DLL/Library
// 2) built with a strong name

using System;
using System.Data;
using Oracle.DataAccess.Client;
using System.EnterpriseServices;

[Transaction(TransactionOption.RequiresNew)]
public class EnlistDistributedTransactionSample_Comp : ServicedComponent
{
    public void DoWork()
    {
        string constr =
            "User Id=scott;Password=tiger;Data Source=oracle;enlist=false";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Enlist in a distributed transaction
        con.EnlistDistributedTransaction((ITransaction)ContextUtil.Transaction);

        // Update EMP table
        OracleCommand cmd = con.CreateCommand();
        cmd.CommandText = "UPDATE emp set sal = sal + .01";
        cmd.ExecuteNonQuery();

        // Commit
        ContextUtil.SetComplete();

        // Dispose OracleConnection object
        con.Dispose();
    }
}
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" on page 1-1
- OracleConnection Class
- OracleConnection Members

GetSessionInfo()

This method returns a new instance of the OracleGlobalization object that represents the globalization settings of the session.

Declaration

```csharp
// C#
public OracleGlobalization GetSessionInfo();
```

Return Value

The newly created OracleGlobalization object.

Example

```csharp
// C#

using System;
using Oracle.DataAccess.Client;

class GetSessionInfoSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();
    }
}
// Get session info from connection object
OracleGlobalization info = con.GetSessionInfo();

// Execute SQL SELECT
OracleCommand cmd = con.CreateCommand();
cmd.CommandText = "select TO_CHAR(hiredate) from emp";
Console.WriteLine("Hire Date [{0}]: {1}",
    info.DateFormat, cmd.ExecuteScalar());

// Update session info
info.DateFormat = "YYYY-MM-DD";
con.SetSessionInfo(info);

// Execute SQL SELECT again
Console.WriteLine("Hire Date [{0}]: {1}",
    info.DateFormat, cmd.ExecuteScalar());

// Clean up
cmd.Dispose();
con.Dispose();

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- OracleConnection Class
- OracleConnection Members

Open

This method opens a connection to an Oracle database.

Declaration
// C#
public void Open();
Implementes
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" on page 1-1
- 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" on page 1-1
- OracleConnection Class
- OracleConnection Members
- "Password Expiration" on page 3-6

PurgeStatementCache
This method flushes the Statement Cache by closing all open cursors on the server, when Statement Caching is enabled.
Declaration

// C#
public void PurgeStatementCache();

Remarks

Flushing the statement cache repetitively results in decreased performance and may negate the performance benefit gained by enabling the statement cache. Statement Caching remains enabled after the call to PurgeStatementCache.

Example

// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class PurgeStatementCacheSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle;" + "Statement Cache Size=20";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand("select * from emp", con);
        cmd.CommandType = CommandType.Text;
        OracleDataReader reader = cmd.ExecuteReader();

        // Purge Statement Cache
        con.PurgeStatementCache();

        // Close and Dispose OracleConnection object
        Console.WriteLine("Statement Cache Flushed");
        con.Close();
        con.Dispose();
    }
}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members
- "Statement Caching" on page 3-21
- ConnectionString on page 4-62

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#
using System;
using Oracle.DataAccess.Client;

class SetSessionInfoSample
{
  static void Main()
  {
    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    // Get session info from connection object
    OracleGlobalization info = con.GetSessionInfo();

    // Execute SQL SELECT
    OracleCommand cmd = con.CreateCommand();
    cmd.CommandText = "select TO_CHAR(hiredate) from emp";
    Console.WriteLine("Hire Date ({0}): {1}", info.DateFormat, cmd.ExecuteScalar());

    // Update session info
    info.DateFormat = "YYYY-MM-DD";
    con.SetSessionInfo(info);

    // Execute SQL SELECT again
    Console.WriteLine("Hire Date ({0}): {1}", info.DateFormat, cmd.ExecuteScalar());

    // Clean up
    cmd.Dispose();
    con.Dispose();
  }
}

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members
OracleConnection Events

OracleConnection events are listed in Table 4–29.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Failover</td>
<td>An event that is triggered when an Oracle failover occurs</td>
</tr>
<tr>
<td>InfoMessage</td>
<td>An event that is triggered for any message or warning sent by the database</td>
</tr>
<tr>
<td>StateChange</td>
<td>An event that is triggered when the connection state changes</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members

Failover

This event is triggered when an Oracle failover occurs.

Declaration

```csharp
// 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" on page 1-1
- OracleConnection Class
- OracleConnection Members
- "OracleFailoverEventArgs Properties" on page 7-6

InfoMessage

This event is triggered for any message or warning sent by the database.

Declaration

```csharp
// C#
public event OracleInfoMessageEventHandler InfoMessage;
```
**OracleConnection Events**

**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" on page 1-1
- OracleConnection Class
- OracleConnection Members
- "OracleInfoMessageEventArgs Properties" on page 4-190

**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" on page 1-1
- OracleConnection Class
- OracleConnection Members
OracleConnection Event Delegates

OracleConnection event delegates are listed in Table 4–30.

Table 4–30  OracleConnection Event Delegates

<table>
<thead>
<tr>
<th>Event Delegate Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleFailoverEventHandler</td>
<td>An event delegate that handles the Failover event</td>
</tr>
<tr>
<td>OracleInfoMessageEventHandler</td>
<td>An event delegate that handles the InfoMessage event</td>
</tr>
<tr>
<td>StateChangeEventHandler</td>
<td>An event delegate that handles the StateChange event</td>
</tr>
</tbody>
</table>

OracleFailoverEventHandler

This event delegate handles the Failover event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members
- "OracleTransaction Class" on page 4-266

OracleInfoMessageEventHandler

This event delegate handles the InfoMessage event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members
- "OracleInfoMessageEventHandler Delegate" on page 4-193

StateChangeEventHandler

This event delegate handles the StateChange event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleConnection Class
- OracleConnection Members
- Microsoft ADO.NET documentation for a description of StateChangeEventHandler
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

<table>
<thead>
<tr>
<th>MarshalByRefObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
</tr>
<tr>
<td>DataAdapter</td>
</tr>
<tr>
<td>DbDataAdapter</td>
</tr>
</tbody>
</table>

**Declaration**

// C#

```csharp
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 following example uses the OracleDataAdapter and the dataset to update the EMP table:

// C#

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

class OracleDataAdapterSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        string cmdstr = 'SELECT empno, sal from emp';

        // Create the adapter with the selectCommand txt and the
        // connection string
        OracleDataAdapter adapter = new OracleDataAdapter(cmdstr, constr);

        // 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 EMP
        // Command when needed
        OracleCommandBuilder builder = new OracleCommandBuilder(adapter);

        // Get the EMP table from the dataset
        DataTable table = dataset.Tables["EMP"];
    }
}
```
// Indicate DataColumn EMPNO is unique
// This is required by the OracleCommandBuilder to update the EMP table
table.Columns["EMPNO"].Unique = true;

// Get the first row from the EMP table
DataRow row = table.Rows[0];

// Update the salary
double sal = double.Parse(row["SAL"].ToString());
row["SAL"] = sal + .01;

// Now update the EMP using the adapter
// The OracleCommandBuilder will create the UpdateCommand for the
// adapter to update the EMP table
adapter.Update(dataset, "EMP");

Console.WriteLine("Row updated successfully");
}

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
■ "Oracle.DataAccess.Client Namespace" on page 1-1
■ 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
<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleDataAdapter Constructors</td>
<td>Instantiates a new instance of OracleDataAdapter class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleDataAdapter Static Methods
OracleDataAdapter static methods are listed in Table 4–32.

Table 4–32 OracleDataAdapter Static Methods
<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleDataAdapter Properties
OracleDataAdapter properties are listed in Table 4–33.

Table 4–33 OracleDataAdapter Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptChangesDuringFill</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>ContinueUpdateOnError</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>DeleteCommand</td>
<td>A SQL statement or stored procedure to delete rows from an Oracle database</td>
</tr>
<tr>
<td>InsertCommand</td>
<td>A SQL statement or stored procedure to insert new rows into an Oracle database</td>
</tr>
<tr>
<td>MissingMappingAction</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>MissingSchemaAction</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>Requery</td>
<td>Determines whether the SelectCommand is reexecuted on the next call to Fill</td>
</tr>
<tr>
<td>SafeMapping</td>
<td>Creates a mapping between column names in the result set to .NET types, to preserve the data</td>
</tr>
<tr>
<td>SelectCommand</td>
<td>A SQL statement or stored procedure that returns a single or multiple result set</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>TableMappings</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>UpdateCommand</td>
<td>A SQL statement or stored procedure to update rows from the DataSet to an Oracle database</td>
</tr>
</tbody>
</table>

OracleDataAdapter Public Methods
OracleDataAdapter public methods are listed in Table 4–34.
### OracleDataAdapter Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>Fill</td>
<td>Adds or refreshes rows in the DataSet to match the data in the Oracle database (Overloaded)</td>
</tr>
<tr>
<td>FillSchema</td>
<td>Inherited from DbDataAdapter</td>
</tr>
<tr>
<td>GetFillParameters</td>
<td>Inherited from DbDataAdapter</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Update</td>
<td>Inherited from DbDataAdapter</td>
</tr>
</tbody>
</table>

### OracleDataAdapter Events

OracleDataAdapter events are listed in **Table 4–35**.

<table>
<thead>
<tr>
<th>Event Delegate Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHandler</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>FillErrorEventHandler</td>
<td>Inherited from DbDataAdapter</td>
</tr>
<tr>
<td>OracleRowUpdatedEventHandler</td>
<td>Event Delegate for the RowUpdated Event</td>
</tr>
<tr>
<td>OracleRowUpdatingEventHandler</td>
<td>Event Delegate for the RowUpdating Event</td>
</tr>
</tbody>
</table>

### OracleDataAdapter Event Delegates

OracleDataAdapter event delegates are listed in **Table 4–36**.

<table>
<thead>
<tr>
<th>Event Delegate Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHandler</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>FillErrorEventHandler</td>
<td>Inherited from DbDataAdapter</td>
</tr>
<tr>
<td>OracleRowUpdatedEventHandler</td>
<td>Event Delegate for the RowUpdated Event</td>
</tr>
<tr>
<td>OracleRowUpdatingEventHandler</td>
<td>Event Delegate for the RowUpdating Event</td>
</tr>
</tbody>
</table>
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" on page 1-1
- 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:

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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:

- *MissingSchemaAction = MissingSchemaAction.Add*

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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:

- *MissingSchemaAction = MissingSchemaAction.Add*

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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:


See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
OracleDataAdapter Static Methods

OracleDataAdapter static methods are listed in Table 4–37.

Table 4–37  OracleDataAdapter Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
OracleDataAdapter Properties

OracleDataAdapter properties are listed in Table 4–38.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptChangesDuringFill</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>ContinueUpdateOnError</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>DeleteCommand</td>
<td>A SQL statement or stored procedure to delete rows from an Oracle database</td>
</tr>
<tr>
<td>InsertCommand</td>
<td>A SQL statement or stored procedure to insert new rows into an Oracle database</td>
</tr>
<tr>
<td>MissingMappingAction</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>MissingSchemaAction</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>Requery</td>
<td>Determines whether the SelectCommand is reexecuted on the next call to Fill</td>
</tr>
<tr>
<td>SafeMapping</td>
<td>Creates a mapping between column names in the result set to .NET types, to preserve the data</td>
</tr>
<tr>
<td>SelectCommand</td>
<td>A SQL statement or stored procedure that returns a single or multiple result set</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>TableMappings</td>
<td>Inherited from DataAdapter</td>
</tr>
<tr>
<td>UpdateCommand</td>
<td>A SQL statement or stored procedure to update rows from the DataSet to an Oracle database</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- `OracleDataAdapter Class`
- `OracleDataAdapter Members`

**InsertCommand**

This property is a SQL statement or stored procedure to insert new rows into an Oracle database.

**Declaration**

```csharp
// 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" on page 1-1
- `OracleDataAdapter Class`
- `OracleDataAdapter Members`

**Requery**

This property determines whether the `SelectCommand` is reexecuted on the next call to `Fill`.

**Declaration**

```csharp
// C#
public Boolean Requery {get; set;}
```

**Property Value**

Returns `true` if the `SelectCommand` is reexecuted on the next call to `Fill`; otherwise, returns `false`. 
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#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class SafeMappingSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";

        // In this SELECT statement, EMPNO, HIREDATE and SALARY must be
        // preserved using safe type mapping.
        string cmdstr = "SELECT EMPNO, ENAME, HIREDATE, SAL FROM EMP";

        // Create the adapter with the selectCommand txt and the connection string
        OracleDataAdapter adapter = new OracleDataAdapter(cmdstr, constr);

        // 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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-51
// 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 EMP 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 EMP table is case-sensitive,
// the safe type mapping column name must also be case-sensitive.
adapter.SafeMapping.Add("EMPNO", typeof(string));
adapter.SafeMapping.Add("SAL", typeof(string));

// Create and fill the DataSet using the EMP
DataSet dataset = new DataSet();
adapter.Fill(dataset, "EMP");

// Get the EMP table from the dataset
DataTable table = dataset.Tables["EMP"];  

// Get the first row from the EMP table
DataRow row = table.Rows[0];

// Print out the row info
Console.WriteLine("EMPNO Column: type = " + row["EMPNO"].GetType() + "; value = " + row["EMPNO"]);
Console.WriteLine("ENAME Column: type = " + row["ENAME"].GetType() + "; value = " + row["ENAME"]);
Console.WriteLine("HIREDATE Column: type = " + row["HIREDATE"].GetType() + "; value = " + row["HIREDATE"]);
Console.WriteLine("SAL Column: type = " + row["SAL"].GetType() + "; value = " + row["SAL"]);

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Safe Type Mapping" on page 3-49

**SelectCommand**

This property is a SQL statement or stored procedure that returns single or multiple result sets.

**Declaration**

```csharp
// 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" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-51

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" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-51
OracleDataAdapter Public Methods

OracleDataAdapter public methods are listed in Table 4–39.

Table 4–39  OracleDataAdapter Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>Fill</td>
<td>Adds or refreshes rows in the DataSet to match the data in the Oracle database (Overloaded)</td>
</tr>
<tr>
<td>FillSchema</td>
<td>Inherited from DbDataAdapter</td>
</tr>
<tr>
<td>GetFillParameters</td>
<td>Inherited from DbDataAdapter</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Update</td>
<td>Inherited from DbDataAdapter</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-51

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" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-51

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" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-51

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" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-51
OracleDataAdapter events are listed in Table 4-40.

Table 4-40 OracleDataAdapter Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>FillError</td>
<td>Inherited from DbDataAdapter</td>
</tr>
<tr>
<td>RowUpdated</td>
<td>This event is raised when row(s) have been updated by the Update() method</td>
</tr>
<tr>
<td>RowUpdating</td>
<td>This event is raised when row data are about to be updated to the database</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members

RowUpdated

This event is raised when row(s) have been updated by the Update() method.

Declaration

```csharp
// 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.

```csharp
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class RowUpdatedSample
{
    // 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);
    }

    // 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);
    }

    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        string cmdstr = "SELECT EMPNO, ENAME, SAL FROM EMP";

        // Create the adapter with the selectCommand txt and the
        // connection string
        OracleDataAdapter adapter = new OracleDataAdapter(cmdstr, constr);

        // Create and fill the DataSet using the EMP
        DataSet dataset = new DataSet();
        adapter.Fill(dataset, "EMP");

        // Get the EMP table from the dataset
        DataTable table = dataset.Tables["EMP"];";
        
        // Indicate DataColumn EMPNO is unique
        // This is required by the OracleCommandBuilder to update the EMP table
        table.Columns["EMPNO"].Unique = true;

        // Get the first row from the EMP table
        DataRow row = table.Rows[0];
```
// Update the salary
double sal = double.Parse(row["SAL"].ToString());
row["SAL"] = sal + .01;

// 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 EMP using the adapter
// The OracleCommandBuilder will create the UpdateCommand for the
// adapter to update the EMP table
// The OnRowUpdating() and the OnRowUpdated() methods will be triggered
adapter.Update(dataset, "EMP");
}
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 4-105.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
OracleDataAdapter Event Delegates

OracleDataAdapter event delegates are listed in Table 4–41.

<table>
<thead>
<tr>
<th>Event Delegate Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHandler</td>
<td>Inherited from Component</td>
</tr>
<tr>
<td>FillErrorEventHandler</td>
<td>Inherited from DbDataAdapter</td>
</tr>
<tr>
<td>OracleRowUpdatedEventHandler</td>
<td>Event Delegate for the RowUpdated Event</td>
</tr>
<tr>
<td>OracleRowUpdatingEventHandler</td>
<td>Event Delegate for the RowUpdating Event</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleRowUpdatedEventHandler

This event delegate handles the RowUpdated Event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- OracleRowUpdatedEventHandler Delegate on page 4-250

OracleRowUpdatingEventHandler

This event delegate handles the RowUpdating Event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleRowUpdatingEventArgs Class" on page 4-258
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.

The following section contain related information:

- "Obtaining LONG and LONG RAW Data" on page 3-27.
- "Obtaining Data From an OracleDataReader" on page 3-23.

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 following OracleDataReader example retrieves the data from the EMP table:

/* Database Setup

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

Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION) values
(1,'KING','SOFTWARE ENGR');
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION) values
(2,'SCOTT','MANAGER');
commit;
*/
using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleDataReaderSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        string cmdstr = "SELECT * FROM EMPINFO";
        OracleConnection connection = new OracleConnection(constr);
        OracleCommand cmd = new OracleCommand(cmdstr, con);

        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();
}

// Clean up
reader.Dispose();
con.Dispose();
}

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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>
<thead>
<tr>
<th>Table 4–42 OracleDataReader Static Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td>Equals</td>
</tr>
</tbody>
</table>

**OracleDataReader Properties**
OracleDataReader properties are listed in Table 4–43.

<table>
<thead>
<tr>
<th>Table 4–43 OracleDataReader Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property</strong></td>
</tr>
<tr>
<td>Depth</td>
</tr>
<tr>
<td>FetchSize</td>
</tr>
<tr>
<td>FieldCount</td>
</tr>
<tr>
<td>HasRows</td>
</tr>
<tr>
<td>IsClosed</td>
</tr>
<tr>
<td>Item</td>
</tr>
<tr>
<td>InitialLOBFetchSize</td>
</tr>
<tr>
<td>InitialLONGFetchSize</td>
</tr>
<tr>
<td>RecordsAffected</td>
</tr>
</tbody>
</table>

**OracleDataReader Public Methods**
OracleDataReader public methods are listed in Table 4–44.

<table>
<thead>
<tr>
<th>Table 4–44 OracleDataReader Public Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Public Method</strong></td>
</tr>
<tr>
<td>Close</td>
</tr>
<tr>
<td>CreateObjRef</td>
</tr>
<tr>
<td>Dispose</td>
</tr>
<tr>
<td>Equals</td>
</tr>
<tr>
<td>GetBoolean</td>
</tr>
<tr>
<td>GetByte</td>
</tr>
<tr>
<td>GetBytes</td>
</tr>
</tbody>
</table>
### Table 4-44 (Cont.) OracleDataReader Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetChar</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetChars</td>
<td>Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column</td>
</tr>
<tr>
<td>GetData</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetDataTypeName</td>
<td>Returns the ODP.NET type name of the specified column</td>
</tr>
<tr>
<td>GetDateTime</td>
<td>Returns the DateTime value of the specified column</td>
</tr>
<tr>
<td>GetDecimal</td>
<td>Returns the decimal value of the specified NUMBER column</td>
</tr>
<tr>
<td>GetDouble</td>
<td>Returns the double value of the specified NUMBER column or BINARY_DOUBLE column</td>
</tr>
<tr>
<td>GetFieldType</td>
<td>Returns the Type of the specified column</td>
</tr>
<tr>
<td>GetFloat</td>
<td>Returns the float value of the specified NUMBER column or BINARY_FLOAT column</td>
</tr>
<tr>
<td>GetGuid</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetInt16</td>
<td>Returns the Int16 value of the specified NUMBER column</td>
</tr>
<tr>
<td>GetInt32</td>
<td>Returns the Int32 value of the specified NUMBER column</td>
</tr>
<tr>
<td>GetInt64</td>
<td>Returns the Int64 value of the specified NUMBER column</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited by MarshalByRefObject</td>
</tr>
<tr>
<td>GetName</td>
<td>Returns the name of the specified column</td>
</tr>
<tr>
<td>GetOracleBFile</td>
<td>Returns an OracleBFile object of the specified BFILE column</td>
</tr>
<tr>
<td>GetOracleBinary</td>
<td>Returns an OracleBinary structure of the specified column</td>
</tr>
<tr>
<td>GetOracleBlob</td>
<td>Returns an OracleBlob object of the specified BLOB column</td>
</tr>
<tr>
<td>GetOracleBlobForUpdate</td>
<td>Returns an updatable OracleBlob object of the specified BLOB column</td>
</tr>
<tr>
<td>GetOracleClob</td>
<td>Returns an OracleClob object of the specified CLOB column</td>
</tr>
<tr>
<td>GetOracleClobForUpdate</td>
<td>Returns an updatable OracleClob object of the specified CLOB column</td>
</tr>
<tr>
<td>GetOracleDate</td>
<td>Returns an OracleDate structure of the specified DATE column</td>
</tr>
<tr>
<td>GetOracleDecimal</td>
<td>Returns an OracleDecimal structure of the specified NUMBER column</td>
</tr>
<tr>
<td>GetOracleIntervalDS</td>
<td>Returns an OracleIntervalDS structure of the specified INTERVAL DAY TO SECOND column</td>
</tr>
<tr>
<td>GetOracleIntervalYM</td>
<td>Returns an OracleIntervalYM structure of the specified INTERVAL YEAR TO MONTH column</td>
</tr>
<tr>
<td>GetOracleString</td>
<td>Returns an OracleString structure of the specified column</td>
</tr>
</tbody>
</table>
### Table 4–44  (Cont.) OracleDataReader Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetOracleTimeStamp</td>
<td>Returns an OracleTimeStamp structure of the Oracle TimeStamp column</td>
</tr>
<tr>
<td>GetOracleTimeStampLTZ</td>
<td>Returns an OracleTimeStampLTZ structure of the specified Oracle TimeStamp WITH LOCAL TIME ZONE column</td>
</tr>
<tr>
<td>GetOracleTimeStampTZ</td>
<td>Returns an OracleTimeStampTZ structure of the specified Oracle TimeStamp WITH TIME ZONE column</td>
</tr>
<tr>
<td>GetOracleXmlType</td>
<td>Returns an OracleXmlType object of the specified XMLType column</td>
</tr>
<tr>
<td>GetOracleValue</td>
<td>Returns the specified column value as a ODP.NET type</td>
</tr>
<tr>
<td>GetOracleValues</td>
<td>Gets all the column values as ODP.NET types</td>
</tr>
<tr>
<td>GetOrdinal</td>
<td>Returns the 0-based ordinal (or index) of the specified column name</td>
</tr>
<tr>
<td>GetSchemaTable</td>
<td>Returns a DataTable that describes the column metadata of the OracleDataReader</td>
</tr>
<tr>
<td>GetString</td>
<td>Returns the string value of the specified column</td>
</tr>
<tr>
<td>GetTimeSpan</td>
<td>Returns the TimeSpan value of the specified INTERVAL DAY TO SECOND column</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object class</td>
</tr>
<tr>
<td>GetValue</td>
<td>Returns the column value as a .NET type</td>
</tr>
<tr>
<td>GetValues</td>
<td>Gets all the column values as .NET types</td>
</tr>
<tr>
<td>GetXmlReader</td>
<td>Returns the contents of an XMLType column as an instance of an .NET XmlTextReader object</td>
</tr>
<tr>
<td>IsDBNull</td>
<td>Indicates whether the column value is null</td>
</tr>
<tr>
<td>NextResult</td>
<td>Advances the data reader to the next result set when reading the results</td>
</tr>
<tr>
<td>Read</td>
<td>Reads the next row in the result set</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
OracleDataReader Static Methods

OracleDataReader static methods are listed in Table 4–45.

Table 4–45  OracleDataReader Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
OracleDataReader Properties

OracleDataReader properties are listed in Table 4–46.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth</td>
<td>Gets a value indicating the depth of nesting for the current row</td>
</tr>
<tr>
<td>FetchSize</td>
<td>Specifies the size of OracleDataReader’s internal cache</td>
</tr>
<tr>
<td>FieldCount</td>
<td>Gets the number of columns in the result set</td>
</tr>
<tr>
<td>HasRows</td>
<td>Indicates whether the OracleDataReader has one or more rows</td>
</tr>
<tr>
<td>IsClosed</td>
<td>Indicates whether the data reader is closed</td>
</tr>
<tr>
<td>Item</td>
<td>Gets the value of the column (Overloaded)</td>
</tr>
<tr>
<td>InitialLOBFetchSize</td>
<td>Specifies the amount that the OracleDataReader initially fetches for LOB columns</td>
</tr>
<tr>
<td>InitialLONGFetchSize</td>
<td>Specifies the amount that the OracleDataReader initially fetches for LONG and LONG RAW columns</td>
</tr>
<tr>
<td>RecordsAffected</td>
<td>Gets the number of rows changed, inserted, or deleted by execution of the SQL statement</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

Depth

This property gets a value indicating the depth of nesting for the current row.

Declaration

```c#
// 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.
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" on page 1-1
▪ OracleDataReader Class
▪ OracleDataReader Members

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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

HasRows
This property indicates whether the OracleDataReader has one or more rows.

Declaration

```csharp
// C#
public bool HasRows {get};
```

Return Value
bool

Remarks
HasRows indicates whether the OracleDataReader has any rows.
The value of HasRows does not change based on the row position. For example, even if the application has read all the rows from the result set and the next Read method invocation will return false, the HasRows property still returns true since the result set was not empty to begin with.

Rows are fetched to determined the emptiness of the OracleDataReader when HasRows property is accessed for the first time after the creation of the OracleDataReader object.

Example

```csharp
// C#

using System;
using Oracle.DataAccess.Client;

class HasRowsSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = new OracleCommand("select * from emp where empno = 9999", con);

        OracleDataReader reader = cmd.ExecuteReader();
    }
}
```
if (!reader.HasRows)
    Console.WriteLine("The result set is empty.");
else
    Console.WriteLine("The result set is not empty.");

con.Dispose();
}
}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
- http://msdn.microsoft.com/library for detailed information about this Microsoft .NET Framework 1.1 feature

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" on page 1-1
- 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.
### Item [index]

This property gets the .NET value of the column specified by the column index.

**Declaration**

```csharp
// 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

### Item [string]

This property gets the .NET value of the column specified by the column name.

**Declaration**

```csharp
// 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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
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" on page 1-1
- 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
- "InitialLOBFetchSize" on page 4-17 for further information on OracleCommand.InitialLOBFetchSize
- "Obtaining LOB Data" on page 3-27

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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
- "InitialLONGFetchSize" on page 4-18 for further information on OracleCommand.InitialLONGFetchSize
- "Obtaining LONG and LONG RAW Data" on page 3-27

**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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

---

4-122 Oracle Data Provider for .NET Developer’s Guide
OracleDataReader Public Methods

OracleDataReader public methods are listed in Table 4–47.

**Table 4–47  OracleDataReader Public Methods**

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Closes the OracleDataReader</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases any resources or memory allocated by the object</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBoolean</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetByte</td>
<td>Returns the byte value of the specified column</td>
</tr>
<tr>
<td>GetBytes</td>
<td>Populates the provided byte array with up to the maximum number of bytes, from the specified offset (in bytes) of the column</td>
</tr>
<tr>
<td>GetChar</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetChars</td>
<td>Populates the provided character array with up to the maximum number of characters, from the specified offset (in characters) of the column</td>
</tr>
<tr>
<td>GetData</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetDataTypeName</td>
<td>Returns the ODP.NET type name of the specified column</td>
</tr>
<tr>
<td>GetDateTime</td>
<td>Returns the DateTime value of the specified column</td>
</tr>
<tr>
<td>GetDecimal</td>
<td>Returns the decimal value of the specified NUMBER column</td>
</tr>
<tr>
<td>GetDouble</td>
<td>Returns the double value of the specified NUMBER column or BINARY_DOUBLE column</td>
</tr>
<tr>
<td>GetFieldType</td>
<td>Returns the Type of the specified column</td>
</tr>
<tr>
<td>GetFloat</td>
<td>Returns the float value of the specified NUMBER column or BINARY_FLOAT column</td>
</tr>
<tr>
<td>GetGuid</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetInt16</td>
<td>Returns the Int16 value of the specified NUMBER column</td>
</tr>
<tr>
<td>GetInt32</td>
<td>Returns the Int32 value of the specified NUMBER column</td>
</tr>
<tr>
<td>GetInt64</td>
<td>Returns the Int64 value of the specified NUMBER column</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited by MarshalByRefObject</td>
</tr>
<tr>
<td>GetName</td>
<td>Returns the name of the specified column</td>
</tr>
<tr>
<td>GetOracleBFile</td>
<td>Returns an OracleBFile object of the specified BFILE column</td>
</tr>
<tr>
<td>GetOracleBinary</td>
<td>Returns an OracleBinary structure of the specified column</td>
</tr>
<tr>
<td>GetOracleBlob</td>
<td>Returns an OracleBlob object of the specified BLOB column</td>
</tr>
<tr>
<td>GetOracleBlobForUpdate</td>
<td>Returns an updatable OracleBlob object of the specified BLOB column</td>
</tr>
<tr>
<td>Public Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetOracleClob</td>
<td>Returns an <code>OracleClob</code> object of the specified CLOB column</td>
</tr>
<tr>
<td>GetOracleClobForUpdate</td>
<td>Returns an updatable <code>OracleClob</code> object of the specified CLOB column</td>
</tr>
<tr>
<td>GetOracleDate</td>
<td>Returns an <code>OracleDate</code> structure of the specified DATE column</td>
</tr>
<tr>
<td>GetOracleDecimal</td>
<td>Returns an <code>OracleDecimal</code> structure of the specified NUMBER column</td>
</tr>
<tr>
<td>GetOracleIntervalDS</td>
<td>Returns an <code>OracleIntervalDS</code> structure of the specified INTERVAL DAY TO SECOND column</td>
</tr>
<tr>
<td>GetOracleIntervalYM</td>
<td>Returns an <code>OracleIntervalYM</code> structure of the specified INTERVAL YEAR TO MONTH column</td>
</tr>
<tr>
<td>GetOracleString</td>
<td>Returns an <code>OracleString</code> structure of the specified column</td>
</tr>
<tr>
<td>GetOracleTimeStamp</td>
<td>Returns an <code>OracleTimeStamp</code> structure of the Oracle TimeStamp column</td>
</tr>
<tr>
<td>GetOracleTimeStampLTZ</td>
<td>Returns an <code>OracleTimeStampLTZ</code> structure of the specified Oracle TimeStamp WITH LOCAL TIME ZONE column</td>
</tr>
<tr>
<td>GetOracleTimeStampTZ</td>
<td>Returns an <code>OracleTimeStampTZ</code> structure of the specified Oracle TimeStamp WITH TIME ZONE column</td>
</tr>
<tr>
<td>GetOracleXmlType</td>
<td>Returns an <code>OracleXmlType</code> object of the specified XMLType column</td>
</tr>
<tr>
<td>GetOracleValue</td>
<td>Returns the specified column value as an ODP.NET type</td>
</tr>
<tr>
<td>GetOracleValues</td>
<td>Gets all the column values as ODP.NET types</td>
</tr>
<tr>
<td>GetOrdinal</td>
<td>Returns the 0-based ordinal (or index) of the specified column name</td>
</tr>
<tr>
<td>GetSchemaTable</td>
<td>Returns a <code>DataTable</code> that describes the column metadata of the <code>OracleDataReader</code></td>
</tr>
<tr>
<td>GetString</td>
<td>Returns the string value of the specified column</td>
</tr>
<tr>
<td>GetTimeSpan</td>
<td>Returns the <code>TimeSpan</code> value of the specified INTERVAL DAY TO SECOND column</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from <code>Object</code> class</td>
</tr>
<tr>
<td>GetValue</td>
<td>Returns the column value as a .NET type</td>
</tr>
<tr>
<td>GetValues</td>
<td>Gets all the column values as .NET types</td>
</tr>
<tr>
<td>GetXmlReader</td>
<td>Returns the value of an XMLType column as an instance of an .NET XmlTextReader</td>
</tr>
<tr>
<td>IsDBNull</td>
<td>Indicates whether the column value is null</td>
</tr>
<tr>
<td>NextResult</td>
<td>Advances the data reader to the next result set when reading the results</td>
</tr>
<tr>
<td>Read</td>
<td>Reads the next row in the result set</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from <code>Object</code></td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 4-109.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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**
  

OracleDataReader Class

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" on page 1-1
- 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

```csharp
// 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

**GetDataTypeName**

This method returns the ODP.NET type name of the specified column.

**Declaration**
```csharp
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" on page 1-1
- 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

GetDecimal

This method returns the decimal value of the specified NUMBER column.

Declaration

// C#
public decimal GetDecimal(int index);
OracleDataReader Public Methods

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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

**GetDouble**

This method returns the double value of the specified NUMBER column or BINARY_DOUBLE column.

Declaration

```csharp
// 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 10g, GetDouble now supports retrieval of data from BINARY_DOUBLE columns.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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.
- ArgumentOutOfRangeException - 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 10g, GetFloat now supports retrieval of data from BINARY_FLOAT columns.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 shortGetInt16(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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
■ 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#

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class GetOracleBlobForUpdateSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Get the ByteCodes for empno = 1
        string cmdstr = "SELECT BYTECODES, EMPNO FROM EMPINFO where EMPNO = 1";
        OracleCommand cmd = new OracleCommand(cmdstr, con);

        // Since we are going to update the OracleBlob object, we will
        // have to create a transaction
        OracleTransaction txn = con.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 ByteCodes object
byte[] addedBytes = new byte[2] { 0, 0 }; byteCodesBlob.Append(addedBytes, 0, addedBytes.Length);

// Now commit the transaction
txn.Commit();
Console.WriteLine("Blob Column successfully updated");
} else
reader.Dispose();

// Close the connection
con.Dispose();
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-32

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 Oracle9i or later. For any version lower than Oracle9i, 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 4-138 for a code example demonstrating usage.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- `OracleDataReader Class`
- `OracleDataReader Members`
- "LOB Support" on page 3-32

GetOracleClob

This method returns an `OracleClob` object of the specified CLOB column.

Declaration

```csharp
// 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" on page 1-1
- `OracleDataReader Class`
- `OracleDataReader Members`
- "LOB Support" on page 3-32

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);
**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.

```csharp
// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class GetOracleClobForUpdateSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Get the job description for empno = 1
        string cmdStr = "SELECT JOBDESCRIPTION, EMPNO FROM EMPINFO where EMPNO = 1";
        OracleCommand cmd = new OracleCommand(cmdStr, con);

        // Since we are going to update the OracleClob object, we will
        // have to create a transaction
        OracleTransaction txn = con.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();
    Console.WriteLine("Clob Column successfully updated");
}
else
    reader.Close();

// Close the connection
con.Close();

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-32

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 Oracle9i or later. For any version lower than Oracle9i, 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 4-142 for a code example demonstrating usage.
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-32

GetOracleDate
This method returns an OracleDate structure of the specified DATE column.

Declaration

```csharp
// 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-32

GetOracleDecimal
This method returns an OracleDecimal structure of the specified NUMBER column.

Declaration

```csharp
// 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

---

### GetOracleIntervalDS

This method returns an **OracleIntervalDS** structure of the specified **INTERVAL DAY TO SECOND** column.

#### Declaration

```csharp
// 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" on page 1-1
- 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" on page 1-1
■ 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.
OracleDataReader Public Methods

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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

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 Oracle9i Release 2 (9.2) or higher.
GetOracleValue

This method returns the specified column value as an ODP.NET type.

**Declaration**

```csharp
// 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

GetOracleValues

This method gets all the column values as ODP.NET types.

**Declaration**

```csharp
// 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-32

GetOrdinal

This method returns the 0-based ordinal (or index) of the specified column name.

Declaration

```csharp
// 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" on page 1-1
- 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>
<thead>
<tr>
<th>Name</th>
<th>Name Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnNameTB</td>
<td>System.String</td>
<td>The name of the column.</td>
</tr>
<tr>
<td>ColumnOrdinal</td>
<td>System.Int32</td>
<td>The 0-based ordinal of the column.</td>
</tr>
<tr>
<td>ColumnSize</td>
<td>System.Int64</td>
<td>The maximum possible length of a value in the column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ColumnSize value is determined as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ CHAR and VARCHAR2 types:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in bytes - if IsByteSemantic boolean value is true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in characters - if IsByteSemantic boolean value is false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ All other types:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in bytes</td>
</tr>
<tr>
<td>NumericPrecision</td>
<td>System.Int16</td>
<td>The maximum precision of the column, if the column is a numeric datatype.</td>
</tr>
<tr>
<td>NumericScale</td>
<td>System.Int16</td>
<td>The scale of the column.</td>
</tr>
</tbody>
</table>

See "IsByteSemantic" on page 4-155 for more information.
### OracleDataReader Public Methods

**Table 4–48  (Cont.) OracleDatareader SchemaTable**

<table>
<thead>
<tr>
<th>Name</th>
<th>Name Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsUnique</td>
<td>System.Boolean</td>
<td>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.</td>
</tr>
<tr>
<td>IsKey</td>
<td>System.Boolean</td>
<td>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.</td>
</tr>
<tr>
<td>IsRowID</td>
<td>System.Boolean</td>
<td>True if the column is a ROWID, otherwise false.</td>
</tr>
<tr>
<td>BaseColumnName</td>
<td>System.String</td>
<td>The name of the column in the database if an alias is used for the column.</td>
</tr>
<tr>
<td>BaseSchemaName</td>
<td>System.String</td>
<td>The name of the schema in the database that contains the column.</td>
</tr>
<tr>
<td>BaseTableName</td>
<td>System.String</td>
<td>The name of the table or view in the database that contains the column.</td>
</tr>
<tr>
<td>DataType</td>
<td>System.RuntimeType</td>
<td>Maps to the common language runtime type.</td>
</tr>
<tr>
<td>ProviderType</td>
<td>Oracle.DataAccess.Client.OracleDbType</td>
<td>The database column type (OracleDbType) of the column.</td>
</tr>
<tr>
<td>AllowDBNull</td>
<td>System.Boolean</td>
<td>True if null values are allowed, otherwise false.</td>
</tr>
</tbody>
</table>
```csharp
using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class GetSchemaTableSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        string cmdstr = "SELECT EMPNO,EMPNAME FROM EMPINFO where EMPNO = 1";
        OracleCommand cmd = new OracleCommand(cmdstr, con);

        //get the reader
        OracleDataReader reader = cmd.ExecuteReader();

        //get the schema table
        DataTable schemaTable = reader.GetSchemaTable();

        //retrieve the first column info.
        DataRow row = schemaTable.Rows[0];
        Console.WriteLine("Column name: " + row["COLUMNNAME"]);
        Console.WriteLine("Precision: " + row["NUMERICPRECISION"]);
        Console.WriteLine("Scale: " + row["NUMERICSCALE"]);

        reader.Close();
        // Close the connection
        con.Close();
    }
}
```

Table 4–48 (Cont.) OracleDataReader SchemaTable

<table>
<thead>
<tr>
<th>Name</th>
<th>Name Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsAliased</td>
<td>System.Boolean</td>
<td>true if the column is an alias; otherwise false.</td>
</tr>
<tr>
<td>IsByteSemantic</td>
<td>System.Boolean</td>
<td>IsByteSemantic is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ true if the ColumnSize value uses bytes semantics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ false if ColumnSize uses character semantics</td>
</tr>
<tr>
<td>IsExpression</td>
<td>System.Boolean</td>
<td>This value is always true when connected to a database version earlier than Oracle9i.</td>
</tr>
<tr>
<td>IsHidden</td>
<td>System.Boolean</td>
<td>true if the column is hidden; otherwise false.</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>System.Boolean</td>
<td>true if the column is read-only; otherwise false.</td>
</tr>
<tr>
<td>IsLong</td>
<td>System.Boolean</td>
<td>true if the column is a LONG, LONG RAW, BLOB, CLOB, or BFILE; otherwise false.</td>
</tr>
</tbody>
</table>
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" on page 1-1
- 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**

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleDataReader Class
- OracleDataReader Members
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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

**GetValue**
This method returns the column value as a .NET type.

**Declaration**
```csharp
// 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" on page 1-1
- 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" on page 1-1
- OracleDataReader Class
- OracleDataReader Members

GetXmlReader

This method returns the contents of an XMLType column as an instance of an .NET XmlTextReader object.
Declaration

```csharp
// 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” on page 1-1
- OracleDataReader Class
- OracleDataReader Members

IsDBNull

This method indicates whether the column value is NULL.

Declaration

```csharp
// 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 4-109.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-1
- `OracleDataReader` Class
- `OracleDataReader Members`

**NextResult**

This method advances the data reader to the next result set.

**Declaration**

```csharp
// 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" on page 1-1
- `OracleDataReader` Class
- `OracleDataReader Members`

**Read**

This method reads the next row in the result set.

**Declaration**

```csharp
// 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 4-109.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- `OracleDataReader Class`
- `OracleDataReader Members`
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#  
using System;  
using System.Data;  
using Oracle.DataAccess.Client;  

class OracleErrorsSample  
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        try
        {
            cmd.CommandText = "insert into notable values (99, 'MyText')";
            cmd.ExecuteNonQuery();
        }
        catch (OracleException ex)
        {
            // Record is not inserted into the database table.
            Console.WriteLine("Record is not inserted into the database table.");

            foreach (OracleError error in ex.Errors)
            {
                Console.WriteLine("Error Message: " + error.Message);
                Console.WriteLine("Error Source: " + error.Source);
            }
        }
    }
}
Requirements

Namespace: `Oracle.DataAccess.Client`  
Assembly: `Oracle.DataAccess.dll`

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleError Properties

OracleError properties are listed in Table 4–50.

Table 4–50 OracleError Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayBindIndex</td>
<td>Specifies the row number of errors that occurred during the Array Bind execution</td>
</tr>
<tr>
<td>DataSource</td>
<td>Specifies the Oracle service name (TNS name) that identifies the Oracle database</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the message describing the error</td>
</tr>
<tr>
<td>Number</td>
<td>Specifies the Oracle error number</td>
</tr>
<tr>
<td>Procedure</td>
<td>Specifies the stored procedure that causes the error</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the name of the data provider that generates the error</td>
</tr>
</tbody>
</table>

OracleError Methods

OracleError methods are listed in Table 4–51.

Table 4–51 OracleError Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string representation of the OracleError</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleError Class
OracleError Static Methods

OracleError static methods are listed in Table 4–52.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleError Class
- OracleError Members
OracleError Properties

OracleError properties are listed in Table 4–53.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayBindIndex</td>
<td>Specifies the row number of errors that occurred during the Array Bind execution</td>
</tr>
<tr>
<td>DataSource</td>
<td>Specifies the Oracle service name (TNS name) that identifies the Oracle database</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the message describing the error</td>
</tr>
<tr>
<td>Number</td>
<td>Specifies the Oracle error number</td>
</tr>
<tr>
<td>Procedure</td>
<td>Specifies the stored procedure that causes the error</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the name of the data provider that generates the error</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- OracleError Class
- OracleError Members
- "Array Binding" on page 3-18

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" on page 1-1
- 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" on page 1-1
- OracleError Class
- OracleError Members

Number
This property specifies the Oracle error number.

Declaration
// C#
public int Number {get;}

Property Value
An int.
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" on page 1-1
- 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" on page 1-1
- OracleError Class
- OracleError Members
OracleError Methods

OracleError methods are listed in Table 4–54.

Table 4–54 OracleError Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string representation of the OracleError</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 Oracle error number: Class.Method name error message stack trace information.

Example

ORA-24333: zero iteration count

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleError Class
- OracleError Members
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#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleErrorCollectionSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        try
        {
            cmd.CommandText = "insert into notable values (99, 'MyText')";
            cmd.ExecuteNonQuery();
        }
        catch (OracleException ex)
        {
            Console.WriteLine("Record is not inserted into the database table.");

            foreach (OracleError error in ex.Errors)
            {
                Console.WriteLine("Error Message: " + error.Message);
                Console.WriteLine("Error Source: " + error.Source);
            }
        }
    }
}
Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleErrorCollection Properties
OracleErrorCollection properties are listed in Table 4–56.

Table 4–56  OracleErrorCollection Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>Count</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>IsSynchronized</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>Item</td>
<td>Inherited from ArrayList</td>
</tr>
</tbody>
</table>

OracleErrorCollection Public Methods
OracleErrorCollection public methods are listed in Table 4–57.

Table 4–57  OracleErrorCollection Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyTo</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleErrorCollection Class
OracleErrorCollection Static Methods

OracleErrorCollection static methods are listed in Table 4–58.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleErrorCollection Class
- OracleErrorCollection Members
OracleErrorCollection Properties

OracleErrorCollection properties are listed in Table 4–59.

Table 4–59 OracleErrorCollection Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>Count</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>IsSynchronized</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>Item</td>
<td>Inherited from ArrayList</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleErrorCollection Class
- OracleErrorCollection Members
OracleErrorCollection Public Methods

OracleErrorCollection public methods are listed in Table 4–60.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyTo</td>
<td>Inherited from ArrayList</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleExceptionSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        try
        {
            cmd.CommandText = "insert into notable values (99, 'MyText');";
            cmd.ExecuteNonQuery();
        }
        catch (OracleException ex)
        {
            Console.WriteLine("Record is not inserted into the database table.");
            Console.WriteLine("Exception Message: " + ex.Message);
            Console.WriteLine("Exception Source: " + ex.Source);
        }
    }
}
Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleException Properties
OracleException properties are listed in Table 4–62.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource</td>
<td>Specifies the TNS name that contains the information for connecting to an Oracle instance</td>
</tr>
<tr>
<td>Errors</td>
<td>Specifies a collection of one or more OracleError objects that contain information about exceptions generated by the Oracle database</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the error messages that occur in the exception</td>
</tr>
<tr>
<td>Number</td>
<td>Specifies the Oracle error number</td>
</tr>
<tr>
<td>Procedure</td>
<td>Specifies the stored procedure that cause the exception</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the name of the data provider that generates the error</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from Exception</td>
</tr>
</tbody>
</table>

OracleException Methods
OracleException methods are listed in Table 4–63.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Sets the serializable info object with information about the exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns the fully qualified name of this exception</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleException Class
OracleException Static Methods

OracleException static methods are listed in Table 4–64.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleException Class
- OracleException Members
OracleException Properties

OracleException properties are listed in Table 4–65.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource</td>
<td>Specifies the TNS name that contains the information for connecting to an Oracle instance</td>
</tr>
<tr>
<td>Errors</td>
<td>Specifies a collection of one or more OracleError objects that contain information about exceptions generated by the Oracle database</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the error messages that occur in the exception</td>
</tr>
<tr>
<td>Number</td>
<td>Specifies the Oracle error number</td>
</tr>
<tr>
<td>Procedure</td>
<td>Specifies the stored procedure that cause the exception</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the name of the data provider that generates the error</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from Exception</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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.
OracleException Properties

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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- OracleException Class
- OracleException Members
OracleException Methods

OracleException methods are listed in Table 4–66.

### Table 4–66   OracleException Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Sets the serializable info object with information about the exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns the fully qualified name of this exception</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleException Class
- OracleException Members

**GetObjectData**

Overrides Exception

This method sets the serializable info object with information about the exception.

**Declaration**

```csharp
// 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" on page 1-1
- 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#
using System;
using Oracle.DataAccess.Client;

class ToStringSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Create an OracleCommand object using the connection object
        OracleCommand cmd = con.CreateCommand();

        try
        {
            cmd.CommandText = "insert into notable values (99, 'MyText')";
            cmd.ExecuteNonQuery();  // This will throw an exception
        }
        catch (OracleException ex)
        {
            Console.WriteLine("Record is not inserted into the database table.");
            Console.WriteLine("ex.ToString() : " + ex.ToString());
        }
    }
}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleException Class
- OracleException 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#
using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class InfoMessageSample
{
  public static void WarningHandler(object src, OracleInfoMessageEventArgs args)
  {
    Console.WriteLine("Source object is: "+src.GetType().Name);
    Console.WriteLine("InfoMessageArgs.Message is "+args.Message);
    Console.WriteLine("InfoMessageArgs.Source is "+args.Source);
  }
  static void Main()
  {
    OracleConnection con = new OracleConnection("User Id=scott;" +
    "Password=tiger;Data Source=oracle;");
    con.Open();
    OracleCommand cmd = con.CreateCommand();
    //Register to the InfoMessageHandler
    cmd.Connection.InfoMessage += new
    OracleInfoMessageEventHandler(WarningHandler);
    cmd.CommandText =
    "create or replace procedure SelectWithNoInto( " +
    " empname in VARCHAR2) AS " +

"BEGIN " +
" select * from emp where ename = empname; " +
"END SelectWithNoInto;";

// Execute the statement that produces a warning
cmd.ExecuteNonQuery();

// Clean up
cmd.Dispose();
con.Dispose();
}
}

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
■ "Oracle.DataAccess.Client Namespace" on page 1-1
■ 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 4–67.

Table 4–67 OracleInfoMessageEventArgs Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleInfoMessageEventArgs Properties
The OracleInfoMessageEventArgs properties are listed in Table 4–68.

Table 4–68 OracleInfoMessageEventArgs Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>Specifies the collection of errors generated by the data source</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the error text generated by the data source</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the name of the object that generated the error</td>
</tr>
</tbody>
</table>

OracleInfoMessageEventArgs Public Methods
The OracleInfoMessageEventArgs methods are listed in Table 4–69.

Table 4–69 OracleInfoMessageEventArgs Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleInfoMessageEventArgs Class
OracleInfoMessageEventArgs Static Methods

The OracleInfoMessageEventArgs static methods are listed in Table 4–70.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members
OracleInfoMessageEventArgs Properties

The OracleInfoMessageEventArgs properties are listed in Table 4–71.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>Specifies the collection of errors generated by the data source</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the error text generated by the data source</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the name of the object that generated the error</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members

Errors

This property specifies the collection of errors generated by the data source.

**Declaration**

```csharp
// C#
public OracleErrorCollection Errors {get;}
```

**Property Value**

The collection of errors.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members

Message

This property specifies the error text generated by the data source.

**Declaration**

```csharp
// C#
public string Message {get;}
```

**Property Value**

The error text.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members
OracleInfoMessageEventArgs Public Methods

The OracleInfoMessageEventArgs methods are listed in Table 4–72.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- "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#
using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleParameterSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleParameter[] prm = new OracleParameter[3];

        // Create OracleParameter objects through OracleParameterCollection
        OracleCommand cmd = con.CreateCommand();

        cmd.CommandText = "select max(empno) from emp";
        int maxno = int.Parse(cmd.ExecuteScalar().ToString());

        prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal, maxno + 10, 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();
    }
}
Console.WriteLine("Record for employee id {0} has been inserted.", maxno);
}
}

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 4–73.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleParameter Constructors</td>
<td>Instantiates a new instance of OracleParameter class (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleParameter Static Methods**
OracleParameter static methods are listed in Table 4–74.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleParameter Properties**
OracleParameter properties are listed in Table 4–75.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayBindSize</td>
<td>Specifies the input or output size of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution</td>
</tr>
<tr>
<td>ArrayBindStatus</td>
<td>Specifies the input or output status of a parameter before or after an Array Bind or PL/SQL Associative Array Bind execution</td>
</tr>
<tr>
<td>CollectionType</td>
<td>Specifies whether the OracleParameter represents a collection, and if so, specifies the collection type</td>
</tr>
<tr>
<td>DbType</td>
<td>Specifies the datatype of the parameter using the Data.DbType enumeration type</td>
</tr>
<tr>
<td>Direction</td>
<td>Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter</td>
</tr>
<tr>
<td>IsNullable</td>
<td>This method is a no-op</td>
</tr>
<tr>
<td>Offset</td>
<td>Specifies the offset to the Value property or offset to the elements in the Value property</td>
</tr>
<tr>
<td>OracleDbType</td>
<td>Specifies the Oracle datatype</td>
</tr>
<tr>
<td>ParameterName</td>
<td>Specifies the name of the parameter</td>
</tr>
<tr>
<td>Precision</td>
<td>Specifies the maximum number of digits used to represent the Value property</td>
</tr>
<tr>
<td>Scale</td>
<td>Specifies the number of decimal places to which Value property is resolved</td>
</tr>
</tbody>
</table>
**OracleParameter Public Methods**

OracleParameter public methods are listed in Table 4–76.

### Table 4–76  OracleParameter Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a shallow copy of an OracleParameter object</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases allocated resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
- "OracleParameterValueCollection Class" on page 4-225

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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
- "OracleParameterCollection Class" on page 4-225

### OracleParameter(string, object)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name and parameter value.

**Declaration**

```csharp
// 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
OracleParameter Class

- 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
- "OracleParameterCollection Class" on page 4-225

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
OracleParameter Constructors

- 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
- "OracleParameterCollection Class" on page 4-225

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

```csharp
// 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:
OracleParameter Class

- 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
- "OracleParameterCollection Class" on page 4-225

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
OracleParameter Constructors

- 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
- "OracleParameterCollection Class" on page 4-225

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:
OracleParameter Class

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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284

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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
**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**

```csharp
// 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
- "OracleParameterCollection Class" on page 4-225
OracleParameter Static Methods

OracleParameter static methods are listed in Table 4–77.

Table 4–77  OracleParameter Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameter Class
- OracleParameter Members
OracleParameter properties are listed in Table 4–78.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayBindSize</td>
<td>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</td>
</tr>
<tr>
<td>ArrayBindStatus</td>
<td>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</td>
</tr>
<tr>
<td>CollectionType</td>
<td>Specifies whether the OracleParameter represents a collection, and if so, specifies the collection type</td>
</tr>
<tr>
<td>DbType</td>
<td>Specifies the datatype of the parameter using the Data.DbType enumeration type</td>
</tr>
<tr>
<td>Direction</td>
<td>Specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter</td>
</tr>
<tr>
<td>IsNullable</td>
<td>This method is a no-op</td>
</tr>
<tr>
<td>Offset</td>
<td>Specifies the offset to the Value property or offset to the elements in the Value property</td>
</tr>
<tr>
<td>OracleDbType</td>
<td>Specifies the Oracle datatype</td>
</tr>
<tr>
<td>ParameterName</td>
<td>Specifies the name of the parameter</td>
</tr>
<tr>
<td>Precision</td>
<td>Specifies the maximum number of digits used to represent the Value property</td>
</tr>
<tr>
<td>Scale</td>
<td>Specifies the number of decimal places to which Value property is resolved</td>
</tr>
<tr>
<td>Size</td>
<td>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</td>
</tr>
<tr>
<td>SourceColumn</td>
<td>Specifies the name of the DataTable Column of the DataSet</td>
</tr>
<tr>
<td>SourceVersion</td>
<td>Specifies the DataRowVersion value to use when loading the Value property of the parameter</td>
</tr>
<tr>
<td>Status</td>
<td>Indicates the status of the execution related to the data in the Value property</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the value of the Parameter</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameter Class
- OracleParameter Members

**ArrayBindSize**

This property specifies the maximum size, in bytes or characters, of the data for each array element transmitted to or from the server. This property is used for 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 properly. The number of elements in ArrayBindSize must be equal to the value specified in the OracleParameter.Size property.

Example

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class ArrayBindSizeSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleParameter[] prm = new OracleParameter[3];

        // Create OracleParameter objects through OracleParameterCollection
        OracleCommand cmd = con.CreateCommand();
        cmd.CommandText = "select max(empno) from emp";
        int maxno = int.Parse(cmd.ExecuteScalar().ToString());

        // Set the ArrayBindCount for Array Binding
        cmd.ArrayBindCount = 2;

        prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal,
            new int[2] {maxno + 10, maxno + 11}, ParameterDirection.Input);
        prm[1] = cmd.Parameters.Add("paramEname", OracleDbType.Varchar2,
            new string[2] {"Client1xxx", "Client2xxx"}, ParameterDirection.Input);
        prm[2] = cmd.Parameters.Add("paramDeptNo", OracleDbType.Decimal,
            new int[2] {10, 10}, ParameterDirection.Input);

        // Set the ArrayBindSize for prm[1]
        // These sizes indicate the maximum size of the elements in Value property
        prm[1].ArrayBindSize = new int[2];
prm[1].ArrayBindSize[0] = 7; // Set ename = "Client1"
prm[1].ArrayBindSize[1] = 7; // Set ename = "Client2"

cmd.CommandText = "insert into emp(empno, ename, deptno) values(:1, :2, :3)"
   cmd.ExecuteNonQuery();

   Console.WriteLine("Record for employee id {0} has been inserted.", maxno + 10);
   Console.WriteLine("Record for employee id {0} has been inserted.", maxno + 11);

   prm[0].Dispose();
   prm[1].Dispose();
   prm[2].Dispose();
   cmd.Dispose();

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "ArrayBindCount" on page 4-12
- "Size" on page 4-217 and "Value" on page 4-221 for more information on binding Associative Arrays
- "ArrayBindStatus" on page 4-212

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

ArgumentNullException - 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "ArrayBindCount" on page 4-12
- "OracleParameterStatus Enumeration" on page 4-284
- "Value" on page 4-221 for more information on binding Associative Arrays
- "ArrayBindSize" on page 4-210

**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" on page 1-1
- 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
OracleParameter Properties

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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "Inference of OracleDbType from DbType" on page 3-13
- "CollectionType" on page 4-213

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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleDbType Enumeration" on page 4-282
- "Inference of DbType from OracleDbType" on page 3-12
- "CollectionType" on page 4-213

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" on page 1-1
- 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" on page 1-1
- OracleParameter Class
- "Value" on page 4-221
- OracleParameter Members

Scale

This property specifies the number of decimal places to which **Value** property is resolved.

Declaration

```csharp
// 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "Value" on page 4-221

Size

This property specifies the maximum size, in bytes or characters, of the data transmitted to or from the server.

Declaration

```csharp
// 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
For PL/SQL Associative Array Bind, Size specifies the maximum number of elements in PL/SQL Associative Array.

If Size is not explicitly set, it is inferred from the actual size of the specified parameter value when binding only for input parameters. Output parameters must have their size defined explicitly.

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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleDbType Enumeration" on page 4-282
- "CollectionType" on page 4-213
- "ArrayBindSize" on page 4-210
- "ArrayBindStatus" on page 4-212
- "Value" on page 4-221

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" on page 1-1
- 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; }
OracleParameter Properties

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" on page 1-1
- 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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 4-284
- "ArrayBindStatus" on page 4-212

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.
InvalidOperationException - 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.

Please see the following for related information:

- Tables in section "Inference of DbType and OracleDbType from Value" on page 3-14
- "ArrayBindCount" on page 4-12
- "ArrayBindSize" on page 4-210
OracleParameter Properties

- "ArrayBindStatus" on page 4-212
- "OracleDbType Enumeration" on page 4-282

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" on page 1-1
- OracleParameter Class
- OracleParameter Members
- "ArrayBindCount" on page 4-12
- "OracleParameterStatus Enumeration" on page 4-284
OracleParameter Public Methods

OracleParameter public methods are listed in Table 4–79.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a shallow copy of an OracleParameter object</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases allocated resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameter Class
- OracleParameter Members

Clone

This method creates a shallow copy of an OracleParameter object.

Declaration

```csharp
// 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

```csharp
// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class CloneSample
{
```
static void Main()
{
    OracleParameter prm1 = new OracleParameter();

    // Prints "prm1.ParameterName = "
    Console.WriteLine("prm1.ParameterName = " + prm1.ParameterName);

    // Set the ParameterName before cloning
    prm1.ParameterName = "MyParam";

    // Clone the OracleParameter
    OracleParameter prm2 = (OracleParameter) prm1.Clone();

    // Prints "prm2.ParameterName = MyParam"
    Console.WriteLine("prm2.ParameterName = " + prm2.ParameterName);

    prm1.Dispose();
    prm2.Dispose();
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleParameterCollectionSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleParameter[] prm = new OracleParameter[3];

        // Create OracleParameter objects through OracleParameterCollection
        OracleCommand cmd = con.CreateCommand();
        cmd.CommandText = "select max(empno) from emp";
        int maxno = int.Parse(cmd.ExecuteScalar().ToString());

        prm[0] = cmd.Parameters.Add("paramEmpno", OracleDbType.Decimal,
                                  maxno + 10, 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();

Console.WriteLine("Record for employee id {0} has been inserted.", maxno + 10);

// Remove all parameters from OracleParameterCollection
cmd.Parameters.Clear();
prm[0].Dispose();
prm[1].Dispose();
prm[2].Dispose();
cmd.Dispose();

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

**Requirements**

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 4–80.

Table 4–80 OracleParameterCollection Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleParameterCollection Properties
OracleParameterCollection properties are listed in Table 4–81.

Table 4–81 OracleParameterCollection Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Specifies the number of OracleParameters in the collection</td>
</tr>
<tr>
<td>Item</td>
<td>Gets and sets the OracleParameter object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleParameterCollection Public Methods
OracleParameterCollection public methods are listed in Table 4–82.

Table 4–82 OracleParameterCollection Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds objects to the collection (Overloaded)</td>
</tr>
<tr>
<td>Clear</td>
<td>Removes all the OracleParameter objects from the collection</td>
</tr>
<tr>
<td>Contains</td>
<td>Indicates whether objects exist in the collection (Overloaded)</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies OracleParameter objects from the collection, starting with the supplied index to the supplied array</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>IndexOf</td>
<td>Returns the index of the objects in the collection (Overloaded)</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts the supplied OracleParameter to the collection at the specified index</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes objects from the collection</td>
</tr>
</tbody>
</table>
### OracleParameterCollection Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveAt</td>
<td>Removes objects from the collection by location (Overloaded)</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameterCollection Class
OracleParameterCollection Static Methods

OracleParameterCollection static methods are listed in Table 4–83.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameterCollection Class
- OracleParameterCollection Members

OracleParameterCollection Class
OracleParameterCollection Properties

OracleParameterCollection properties are listed in Table 4–84.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Specifies the number of OracleParameters in the collection</td>
</tr>
<tr>
<td>Item</td>
<td>Gets and sets the OracleParameter object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameterCollection Class
- OracleParameterCollection Members

Count

This property specifies the number of OracleParameter objects in the collection.

Declaration

```csharp
// C#
public int Count {get;}
```

Property Value
The number of OracleParameter objects.

Implements
IICollection

Remarks
Default = 0

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- OracleParameterCollection Class
- OracleParameterCollection Members

**Item[int]**
This property gets and sets the `OracleParameter` object at the index specified by the supplied `parameterIndex`.

**Declaration**
```csharp
// 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.

**Item[string]**
This property gets and sets the `OracleParameter` object using the parameter name specified by the supplied `parameterName`.

**Declaration**
```csharp
// C#
public OracleParameter Item[string parameterName] {get; set;};
```

**Property Value**
An `OracleParameter`.

**Implements**
IDataParameterCollection

**Exceptions**
IndexOutOfRangeException - The supplied parameter name does not exist.
OracleParameterCollection Properties

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameterCollection Class
- OracleParameterCollection Members
OracleParameterCollection Public Methods

OracleParameterCollection public methods are listed in Table 4–85.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds objects to the collection (Overloaded)</td>
</tr>
<tr>
<td>Clear</td>
<td>Removes all the OracleParameter objects from the collection</td>
</tr>
<tr>
<td>Contains</td>
<td>Indicates whether objects exist in the collection (Overloaded)</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies OracleParameter objects from the collection, starting with the supplied index to the supplied array</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>IndexOf</td>
<td>Returns the index of the objects in the collection (Overloaded)</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts the supplied OracleParameter to the collection at the specified index</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes objects from the collection</td>
</tr>
<tr>
<td>RemoveAt</td>
<td>Removes objects from the collection by location (Overloaded)</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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.

**Add(object)**

This method adds the supplied object to the collection.

**Declaration**

```csharp
// C#
public int Add(object obj);
```

**Parameters**

- **obj**
  Specifies the supplied object.

**Return Value**

The index at which the new OracleParameter is added.
OracleParameterCollection Class

Implements
IList

Remarks
InvalidCastException - The supplied obj cannot be cast to an OracleParameter object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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.
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" on page 1-1
- 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.
OracleParameterCollection Class

Return Value
The newly created OracleParameter object which was added to the collection.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleParameterCollection Class
- OracleParameterCollection Members
- "OracleDbType Enumeration" on page 4-282

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#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class AddSample
{
    static void Main()
    {
        OracleCommand cmd = new OracleCommand();

        // Add parameter to the OracleParameterCollection
        OracleParameter prm = cmd.Parameters.Add(
            "MyParam", OracleDbType.Decimal, 1, ParameterDirection.Input);

        // Prints "cmd.Parameters.Count = 1"
        Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);
    
}
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" on page 1-1
- OracleParameterCollection Class
- OracleParameterCollection Members
- "OracleDbType Enumeration" on page 4-282

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#

using System;
using Oracle.DataAccess.Client;

class AddSample
{
  static void Main()
  {
    OracleCommand cmd = new OracleCommand();

    // Add parameter to the OracleParameterCollection
    OracleParameter prm = cmd.Parameters.Add("MyParam", OracleDbType.VarChar2, 10);

    // Prints "cmd.Parameters.Count = 1"
    Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);

    prm.Dispose();
    cmd.Dispose();
  }
}

See Also:

■ ["Oracle.DataAccess.Client Namespace" on page 1-1]
■ 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);
OracleParameterCollection Public Methods

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" on page 1-1
- 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" on page 1-1
- `OracleParameterCollection Class`
- `OracleParameterCollection Members`

---

**Clear**
This method removes all the `OracleParameter` objects from the collection.

**Declaration**

```
// C#
public void Clear();
```

**Implements**
`IList`

**Example**

```
// C#
using System;
using Oracle.DataAccess.Client;

class ClearSample
{
    static void Main()
    {
        OracleCommand cmd = new OracleCommand();

        // Add parameter to the OracleParameterCollection
        OracleParameter prm = cmd.Parameters.Add("MyParam", OracleDbType.Decimal);

        // Prints 'cmd.Parameters.Count = 1'
        Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);

        // Clear all parameters in the OracleParameterCollection
        cmd.Clear();
    }
}
```
cmd.Parameters.Clear();

// Prints "cmd.Parameters.Count = 0"
Console.WriteLine("cmd.Parameters.Count = " + cmd.Parameters.Count);

prm.Dispose();
cmd.Dispose();
}
}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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
// C#

using System;
using Oracle.DataAccess.Client;

class ContainsSample
{
    static void Main()
    {
        OracleCommand cmd = new OracleCommand();

        // Add parameter to the OracleParameterCollection
        OracleParameter prm1 = cmd.Parameters.Add("MyParam", OracleDbType.Decimal);

        // Check if the OracleParameterCollection contains prm1
        bool bContains = cmd.Parameters.Contains(prm1);

        // Prints "bContains = True"
        Console.WriteLine("bContains = " + bContains);

        OracleParameter prm2 = new OracleParameter();

        // Check if the OracleParameterCollection contains prm2
        bContains = cmd.Parameters.Contains(prm2);

        // Prints "bContains = False"
        Console.WriteLine("bContains = " + bContains);

        prm1.Dispose();
        prm2.Dispose();
        cmd.Dispose();
    }
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

// C#

using System;
using Oracle.DataAccess.Client;

class ContainsSample
{
  static void Main()
  {
    OracleCommand cmd = new OracleCommand();

    // Add parameter to the OracleParameterCollection
    OracleParameter prm = cmd.Parameters.Add("MyParam", OracleDbType.Decimal);

    // Check if the OracleParameterCollection contains "MyParam"
    bool bContains = cmd.Parameters.Contains("MyParam");

    // Prints "bContains = True"
    Console.WriteLine("bContains = " + bContains);

    // Check if the OracleParameterCollection contains "NoParam"
    bContains = cmd.Parameters.Contains("NoParam");

    // Prints "bContains = False"
    Console.WriteLine("bContains = " + bContains);

    prm.Dispose();
    cmd.Dispose();
  }
}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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.
OracleParameterCollection Public Methods

**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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- **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**

```
// C#

using System;
using Oracle.DataAccess.Client;

class RemoveSample
{
    static void Main()
    {
        OracleCommand cmd = new OracleCommand();

        // Add 2 parameters to the OracleParameterCollection
        OracleParameter prm1 = cmd.Parameters.Add("MyParam1", OracleDbType.Decimal);
        OracleParameter prm2 = cmd.Parameters.Add("MyParam2", OracleDbType.Decimal);

        // Prints "cmd.Parameters.Count = 2"
Console.WriteLine("cmd.Parameters.Count = "+cmd.Parameters.Count);

// Remove the 1st parameter from the OracleParameterCollection
cmd.Parameters.RemoveAt(0);

// Prints "cmd.Parameters.Count = 1"
Console.WriteLine("cmd.Parameters.Count = "+cmd.Parameters.Count);

// Prints "cmd.Parameters[0].ParameterName = MyParam2"
Console.WriteLine("cmd.Parameters[0].ParameterName = "+cmd.Parameters[0].ParameterName);

prm1.Dispose();
prm2.Dispose();
cmd.Dispose();
}
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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
OracleParameterCollection Class

Specifies the index from which the OracleParameter is to be removed.

**Implements**

IList

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- "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 4-105.

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 4–86.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleRowUpdatedEventArgs Constructor</td>
<td>Instantiates a new instance of OracleRowUpdatedEventArgs class</td>
</tr>
</tbody>
</table>

**OracleRowUpdatedEventArgs Static Methods**
The OracleRowUpdatedEventArgs static methods are listed in Table 4–87.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleRowUpdatedEventArgs Properties**
The OracleRowUpdatedEventArgs properties are listed in Table 4–88.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Specifies the OracleCommand that is used when OracleDataAdapter.Update() is called</td>
</tr>
<tr>
<td>Errors</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>RecordsAffected</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>Row</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>StatementType</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>Status</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>TableMapping</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
</tbody>
</table>

**OracleRowUpdatedEventArgs Public Methods**
The OracleRowUpdatedEventArgs properties are listed in Table 4–89.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1

  OracleRowUpdatedEventArgs Class

  OracleRowUpdatedEventArgs Members
OracleRowUpdatedEventArgs Static Methods

The OracleRowUpdatedEventArgs static methods are listed in Table 4–90.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members
OracleRowUpdatedEventArgs Properties

The OracleRowUpdatedEventArgs properties are listed in Table 4-91.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Specifies the OracleCommand that is used when OracleDataAdapter.Update() is called</td>
</tr>
<tr>
<td>Errors</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>RecordsAffected</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>Row</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>StatementType</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>Status</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
<tr>
<td>TableMapping</td>
<td>Inherited from RowUpdatedEventArgs</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members

Command

This property specifies the OracleCommand that is used when OracleDataAdapter.Update() is called.

Declaration

```csharp
// C#
public new OracleCommand Command {get;}
```

Property Value

The OracleCommand executed when Update is called.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members
OracleRowUpdatedEventArgs Public Methods

The OracleRowUpdatedEventArgs properties are listed in Table 4–92.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 4-105.

Requirements

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- `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 4–93.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleRowUpdatingEventArgs Constructor</td>
<td>Instantiates a new instance of OracleRowUpdatingEventArgs class (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleRowUpdatingEventArgs Static Methods**
The OracleRowUpdatingEventArgs static methods are listed in Table 4–94.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleRowUpdatingEventArgs Properties**
The OracleRowUpdatingEventArgs properties are listed in Table 4–95.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Specifies the OracleCommand that is used when the OracleDataAdapter.Update() is called</td>
</tr>
<tr>
<td>Errors</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
<tr>
<td>Row</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
<tr>
<td>StatementType</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
<tr>
<td>Status</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
<tr>
<td>TableMapping</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
</tbody>
</table>

**OracleRowUpdatingEventArgs Public Methods**
The OracleRowUpdatingEventArgs public methods are listed in Table 4–96.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members
OracleRowUpdatingEventArgs Static Methods

The OracleRowUpdatingEventArgs static methods are listed in Table 4–97.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members
OracleRowUpdatingEventArgs Properties

The OracleRowUpdatingEventArgs properties are listed in Table 4–98.

Table 4–98  OracleRowUpdatingEventArgs Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Specifies the OracleCommand that is used when the OracleDataAdapter.Update() is called</td>
</tr>
<tr>
<td>Errors</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
<tr>
<td>Row</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
<tr>
<td>StatementType</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
<tr>
<td>Status</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
<tr>
<td>TableMapping</td>
<td>Inherited from RowUpdatingEventArgs</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members
OracleRowUpdatingEventArgs Public Methods

The OracleRowUpdatingEventArgs public methods are listed in Table 4–99.

Table 4–99 OracleRowUpdatingEventArgs Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members
OracleRowUpdatingEventHandler Delegate

The `OracleRowUpdatingEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdating` event.

**Declaration**

```csharp
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" on page 1-1
- "OracleDataAdapter Class" on page 4-86
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 savepoints 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**
// Database Setup
/*
DROP TABLE MyTable;
CREATE TABLE MyTable (MyColumn NUMBER);
--CREATE TABLE MyTable (MyColumn NUMBER PRIMARY KEY);
*/

// C#
using System;
using System.Data;
using Oracle.DataAccess.Client;

class OracleTransactionSample
{
    static void Main()
string constr = "User Id=scott;Password=tiger;Data Source=oracle";
OracleConnection con = new OracleConnection(constr);
con.Open();

OracleCommand cmd = con.CreateCommand();

// Check the number of rows in MyTable before transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
int myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Print the number of rows in MyTable
Console.WriteLine("myTableCount = " + myTableCount);

// Start a transaction
OracleTransaction txn = con.BeginTransaction(
      IsolationLevel.ReadCommitted);

try
{
    // Insert the same row twice into MyTable
    cmd.CommandText = "INSERT INTO MyTable VALUES (1)";
    cmd.ExecuteNonQuery();
    cmd.ExecuteNonQuery(); // This may throw an exception
    txn.Commit();
}
catch (Exception e)
{
    // Print the exception message
    Console.WriteLine("e.Message = " + e.Message);

    // Rollback the transaction
    txn.Rollback();
}

// Check the number of rows in MyTable after transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Prints the number of rows
// If MyColumn is not a PRIMARY KEY, the value should increase by two.
// If MyColumn is a PRIMARY KEY, the value should remain same.
Console.WriteLine("myTableCount = " + myTableCount);

txn.Dispose();
cmd.Dispose();
con.Close();
con.Dispose();
}
Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 4–100.

Table 4–100 OracleTransaction Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTransaction Properties
OracleTransaction properties are listed in Table 4–101.

Table 4–101 OracleTransaction Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsolationLevel</td>
<td>Specifies the isolation level for the transaction</td>
</tr>
<tr>
<td>Connection</td>
<td>Specifies the connection that is associated with the transaction</td>
</tr>
</tbody>
</table>

OracleTransaction Public Methods
OracleTransaction public methods are listed in Table 4–102.

Table 4–102 OracleTransaction Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>Commits the database transaction</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Frees the resources used by the OracleTransaction object</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Rollback</td>
<td>Rolls back a database transaction (Overloaded)</td>
</tr>
<tr>
<td>Save</td>
<td>Creates a savepoint within the current transaction</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleTransaction Class
OracleTransaction Static Methods

OracleTransaction static methods are listed in Table 4–103.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleTransaction Class
- OracleTransaction Members
OracleTransaction Properties

OracleTransaction properties are listed in Table 4-104.

Table 4–104  OracleTransaction Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsolationLevel</td>
<td>Specifies the isolation level for the transaction</td>
</tr>
<tr>
<td>Connection</td>
<td>Specifies the connection that is associated with the transaction</td>
</tr>
</tbody>
</table>

See Also:
- “Oracle.DataAccess.Client Namespace” on page 1-1
- 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” on page 1-1
- 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" on page 1-1
- OracleTransaction Class
- OracleTransaction Members
OracleTransaction Class

OracleTransaction Public Methods

OracleTransaction public methods are listed in Table 4–105.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>Commits the database transaction</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Frees the resources used by the OracleTransaction object</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Rollback</td>
<td>Rolls back a database transaction (Overloaded)</td>
</tr>
<tr>
<td>Save</td>
<td>Creates a savepoint within the current transaction</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

// Database Setup
/*
DROP TABLE MyTable;
CREATE TABLE MyTable (MyColumn NUMBER);
--CREATE TABLE MyTable (MyColumn NUMBER PRIMARY KEY);
*/

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class CommitSample
{
    static void Main()
    {
        // Drop & Create MyTable as indicated above in Database Setup
        // This sample starts a transaction and inserts two records with the same
        // value for MyColumn into MyTable.
        // If MyColumn is not a primary key, the transaction will commit.
        // If MyColumn is a primary key, the second insert will violate the
        // unique constraint and the transaction will rollback.

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();
        OracleCommand cmd = con.CreateCommand();
        // Check the number of rows in MyTable before transaction
        cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
        int myTableCount = int.Parse(cmd.ExecuteScalar().ToString());
        // Print the number of rows in MyTable
        Console.WriteLine("myTableCount = " + myTableCount);
        // Start a transaction
        OracleTransaction txn = con.BeginTransaction(
            IsolationLevel.ReadCommitted);
        try
        {
            // Insert the same row twice into MyTable
            cmd.CommandText = "INSERT INTO MyTable VALUES (1)"
            cmd.ExecuteNonQuery();
            // This may throw an exception
            txn.Commit();
        }
        catch (Exception e)
        {
            // Print the exception message
            Console.WriteLine("e.Message = " + e.Message);
            // Rollback the transaction
            txn.Rollback();
        }
        // Check the number of rows in MyTable after transaction
        cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
        myTableCount = int.Parse(cmd.ExecuteScalar().ToString());
// Prints the number of rows
// If MyColumn is not a PRIMARY KEY, the value should increase by two.
// If MyColumn is a PRIMARY KEY, the value should remain same.
Console.WriteLine("myTableCount = " + myTableCount);

txn.Dispose();
cmd.Dispose();

con.Close();
con.Dispose();

}
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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.
Rollback()

This method rolls back a database transaction.

**Declaration**

```csharp
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**

```csharp
// Database Setup
/*
DROP TABLE MyTable;
CREATE TABLE MyTable (MyColumn NUMBER);
*/

using System;
using System.Data;
using Oracle.DataAccess.Client;

class RollbackSample
{
    static void Main()
    {
        // Drop & Create MyTable as indicated above in Database Setup
        // This sample starts a transaction and inserts one record into MyTable.
        // It then rollsback the transaction, the number of rows remains the same

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleCommand cmd = con.CreateCommand();

        // Check the number of rows in MyTable before transaction
        cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
    }
}
```
```csharp
int myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Print the number of rows in MyTable
Console.WriteLine("myTableCount = " + myTableCount);

// Start a transaction
OracleTransaction txn = con.BeginTransaction(
    IsolationLevel.ReadCommitted);

// Insert a row into MyTable
cmd.CommandText = "INSERT INTO MyTable VALUES (1)";
cmd.ExecuteNonQuery();

// Rollback the transaction
txn.Rollback();

// Check the number of rows in MyTable after transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Prints the number of rows, should remain the same
Console.WriteLine("myTableCount = " + myTableCount);

txn.Dispose();
cmd.Dispose();
con.Close();
con.Dispose();
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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
// Database Setup
/*
DROP TABLE MyTable;
CREATE TABLE MyTable (MyColumn NUMBER);
*/

// C#

using System;
using System.Data;
using Oracle.DataAccess.Client;

class SaveSample
{
    static void Main()
    {
        // Drop & Create MyTable as indicated above in Database Setup
// This sample starts a transaction and creates a savepoint after
// inserting one record into MyTable.
// After inserting the second record it rolls back to the savepoint
// and commits the transaction. Only the first record will be inserted

string constr = "User Id=scott;Password=tiger;Data Source=oracle";
OracleConnection con = new OracleConnection(constr);
con.Open();

OracleCommand cmd = con.CreateCommand();

// Check the number of rows in MyTable before transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
int myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Print the number of rows in MyTable
Console.WriteLine("myTableCount = " + myTableCount);

// Start a transaction
OracleTransaction txn = con.BeginTransaction(IsolationLevel.ReadCommitted);

// Insert a row into MyTable
cmd.CommandText = "INSERT INTO MyTable VALUES (1)";
cmd.ExecuteNonQuery();

// Create a savepoint
txn.Save("MySavePoint");

// Insert another row into MyTable
cmd.CommandText = "INSERT INTO MyTable VALUES (2)";
cmd.ExecuteNonQuery();

// Rollback to the savepoint
txn.Rollback("MySavePoint");

// Commit the transaction
txn.Commit();

// Check the number of rows in MyTable after transaction
cmd.CommandText = "SELECT COUNT(*) FROM MyTable";
myTableCount = int.Parse(cmd.ExecuteScalar().ToString());

// Prints the number of rows, should have increased by 1
Console.WriteLine("myTableCount = " + myTableCount);

txn.Dispose();
cmd.Dispose();

con.Close();
con.Dispose();
}
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleTransaction Class
- OracleTransaction Members
OracleCollectionType Enumeration

OracleCollectionType enumerated values specify whether the OracleParameter object represents a collection, and if so, specifies the collection type.

Table 4–106 lists all the OracleCollectionType enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Is not a collection type</td>
</tr>
<tr>
<td>PLSQLAssociativeArray</td>
<td>Indicates that the collection type is a PL/SQL Associative Array (or PL/SQL Index-By Table)</td>
</tr>
</tbody>
</table>

See Also:
- “Oracle.DataAccess.Client Namespace” on page 1-1
- "OracleParameter Class” on page 4-194
- "CollectionType“ on page 4-213
OracleDbType Enumeration

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

Table 4–107 lists all the OracleDbType enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFile</td>
<td>Oracle BFILE type</td>
</tr>
<tr>
<td>Blob</td>
<td>Oracle BLOB type</td>
</tr>
<tr>
<td>Byte</td>
<td>byte type</td>
</tr>
<tr>
<td>Char</td>
<td>Oracle CHAR type</td>
</tr>
<tr>
<td>Clob</td>
<td>Oracle CLOB type</td>
</tr>
<tr>
<td>Date</td>
<td>Oracle DATE type</td>
</tr>
<tr>
<td>Decimal</td>
<td>Oracle NUMBER type</td>
</tr>
<tr>
<td>Double</td>
<td>8-byte FLOAT type</td>
</tr>
<tr>
<td>Int16</td>
<td>2-byte INTEGER type</td>
</tr>
<tr>
<td>Int32</td>
<td>4-byte INTEGER type</td>
</tr>
<tr>
<td>Int64</td>
<td>8-byte INTEGER type</td>
</tr>
<tr>
<td>IntervalDS</td>
<td>Oracle INTERVAL DAY TO SECOND type</td>
</tr>
<tr>
<td>IntervalYM</td>
<td>Oracle INTERVAL YEAR TO MONTH type</td>
</tr>
<tr>
<td>Long</td>
<td>Oracle LONG type</td>
</tr>
<tr>
<td>LongRaw</td>
<td>Oracle LONG RAW type</td>
</tr>
<tr>
<td>NChar</td>
<td>Oracle NCHAR type</td>
</tr>
<tr>
<td>NClob</td>
<td>Oracle NCLOB type</td>
</tr>
<tr>
<td>NVarchar2</td>
<td>Oracle NVARCHAR2 type</td>
</tr>
<tr>
<td>Raw</td>
<td>Oracle RAW type</td>
</tr>
<tr>
<td>RefCursor</td>
<td>Oracle REF CURSOR type</td>
</tr>
<tr>
<td>Single</td>
<td>4-byte FLOAT type</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Oracle TIMESTAMP type</td>
</tr>
<tr>
<td>TimestampLTZ</td>
<td>Oracle TIMESTAMP WITH LOCAL TIME ZONE type</td>
</tr>
<tr>
<td>TimestampTZ</td>
<td>Oracle TIMESTAMP WITH TIME ZONE type</td>
</tr>
<tr>
<td>Varchar2</td>
<td>Oracle VARCHAR2 type</td>
</tr>
<tr>
<td>XmlType</td>
<td>Oracle XMLType type</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- "OracleParameter Class" on page 4-194
- "OracleParameterCollection Class" on page 4-225
- OracleParameter "OracleDbType" on page 4-215
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 4–108 lists all the OracleParameterStatus enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>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.</td>
</tr>
<tr>
<td>NullFetched</td>
<td>Indicates that a NULL value has been fetched from a column or an OUT parameter</td>
</tr>
<tr>
<td>NullInsert</td>
<td>Indicates that a NULL value is to be inserted into a column</td>
</tr>
<tr>
<td>Truncation</td>
<td>Indicates that truncation has occurred when fetching the data from the column</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- "OracleParameter Class" on page 4-194
- OracleParameter "ArrayBindStatus" on page 4-212
- OracleParameter "Value" on page 4-221
This chapter describes ODP.NET XML-related classes and enumerations.

This chapter contains these topics:
- OracleXmlCommandType Enumeration
- OracleXmlQueryProperties Class
- OracleXmlSaveProperties Class
- OracleXmlStream Class
- OracleXmlType Class

All offsets are 0-based for OracleXmlStream object parameters.
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 5-1 lists all the OracleXmlCommandType enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>No XML operation is desired</td>
</tr>
<tr>
<td>Query</td>
<td>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</td>
</tr>
<tr>
<td>Insert</td>
<td>The command text is an XML document containing rows to insert.</td>
</tr>
<tr>
<td>Update</td>
<td>The command text is an XML document containing rows to update.</td>
</tr>
<tr>
<td>Delete</td>
<td>The command text is an XML document containing rows to delete.</td>
</tr>
</tbody>
</table>

See Also: "Oracle.DataAccess.Client Namespace" on page 1-1
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#
using System;
using Oracle.DataAccess.Client;

class OracleXmlSavePropertiesSample
{
    static void Main()
    {
        string[] KeyColumnsList = null;
        string[] UpdateColumnsList = null;
        int rows = 0;

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

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

        // Set the XML command type to insert.
        cmd.Cmd.CommandType = OracleXmlCommandType.Insert;

        // Set the XML document.
cmd.CommandText = '{@xml version="1.0"?>
"<ROWSET>
"<MYROW num = "1">
"<EMPLOYEE_ID>1234</EMPLOYEE_ID>
"<LAST_NAME>Smith</LAST_NAME>
"<EMAIL>Smith@Oracle.com</EMAIL>
"<HIRE_DATE>1982-01-23T00:00:00.000</HIRE_DATE>
"<JOB_ID>IT_PROG</JOB_ID>
"</MYROW>
"<MYROW num = "2">
"<EMPLOYEE_ID>1235</EMPLOYEE_ID>
"<LAST_NAME>Barney</LAST_NAME>
"<EMAIL>Barney@Oracle.com</EMAIL>
"<HIRE_DATE>1982-01-23T00:00:00.000</HIRE_DATE>
"<JOB_ID>IT_PROG</JOB_ID>
"</MYROW>
"</ROWSET>";

// 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";

// Do the inserts.
rows = cmd.ExecuteNonQuery();

// Set the XML command type to update.
cmd.XmlCommandType = OracleXmlCommandType.Update;

// Set the XML document.
cmd.CommandText = '{@xml version="1.0"?>
"<ROWSET>
"<MYROW num = "1">
"<EMPLOYEE_ID>1234</EMPLOYEE_ID>
"<LAST_NAME>Adams</LAST_NAME>
"</MYROW>
"</ROWSET>";

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";
UpdateColumnsList = new string[1];
UpdateColumnsList[0] = "LAST_NAME";

// Call the ExecuteNonQuery method.
rows = cmd.ExecuteNonQuery();

// Write the result to the console.
Console.WriteLine("rows: " + rows);'}
// Set the XML command type to delete.
cmd.XmlCommandType = OracleXmlCommandType.Delete;

// Set the XML document.
// cmd.CommandText = "<ROWSET>
//   <MYROW num = \"1\">\n//     <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n//   </MYROW>
//   <MYROW num = \"2\">\n//     <EMPLOYEE_ID>1235</EMPLOYEE_ID>\n//   </MYROW>
// </ROWSET>";

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";
keyColumnsList = KeyColumnsList;
KeyColumnsList.KeyColumnsList = KeyColumnsList;
KeyColumnsList.UpdateColumnsList = null;

// Do the deletes.
rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Clean up.
con.Dispose();
con.Close();
con.Dispose();
}
})

Requirements

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleXmlQueryProperties Members
- OracleXmlQueryProperties Constructor
- OracleXmlQueryProperties Properties
- OracleXmlQueryProperties Public Methods
OracleXmlQueryProperties Members

OracleXmlQueryProperties members are listed in the following tables:

OracleXmlQueryProperties Constructors
The OracleXmlQueryProperties constructors are listed in Table 5–2.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleXmlQueryProperties Constructor</td>
<td>Instantiates a new instance of the OracleXmlQueryProperties class</td>
</tr>
</tbody>
</table>

OracleXmlQueryProperties Properties
The OracleXmlQueryProperties properties are listed in Table 5–3.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxRows</td>
<td>Specifies the maximum number of rows from the result set of the query that can be represented in the result XML document</td>
</tr>
<tr>
<td>RootTag</td>
<td>Specifies the root element of the result XML document</td>
</tr>
<tr>
<td>RowTag</td>
<td>Specifies the value of the XML element which identifies a row of data from the result set in an XML document</td>
</tr>
<tr>
<td>Xslt</td>
<td>Specifies the XSL document used for XML transformation using XSLT</td>
</tr>
<tr>
<td>XsltParams</td>
<td>Specifies parameters for the XSL document</td>
</tr>
</tbody>
</table>

OracleXmlQueryProperties Public Methods
The OracleXmlQueryProperties public methods are listed in Table 5–4.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleXmlQueryProperties object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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' on page 1-1
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
OracleXmlQueryProperties Properties

The OracleXmlQueryProperties properties are listed in Table 5–5.

Table 5–5  OracleXmlQueryProperties Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxRows</td>
<td>Specifies the maximum number of rows from the result set of the query that can be represented in the result XML document</td>
</tr>
<tr>
<td>RootTag</td>
<td>Specifies the root element of the result XML document</td>
</tr>
<tr>
<td>RowTag</td>
<td>Specifies the value of the XML element which identifies a row of data from the result set in an XML document</td>
</tr>
<tr>
<td>Xslt</td>
<td>Specifies the XSL document used for XML transformation using XSLT</td>
</tr>
<tr>
<td>XsltParams</td>
<td>Specifies parameters for the XSL document</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members

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" on page 1-1
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
**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” on page 1-1
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members

**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” on page 1-1
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members

**Xslt**

This property specifies the XSL document used for XML transformation using XSLT.
OracleXmlQueryProperties Properties

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" on page 1-1
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members

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" on page 1-1
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
OracleXmlQueryProperties Public Methods

The OracleXmlQueryProperties public methods are listed in Table 5–6.

Table 5–6 OracleXmlQueryProperties Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleXmlQueryProperties object</td>
</tr>
</tbody>
</table>

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" on page 1-1
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
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#
using System;
using Oracle.DataAccess.Client;

class OracleXmlSavePropertiesSample
{
    static void Main()
    {
        string[] KeyColumnsList = null;
        string[] UpdateColumnsList = null;
        int rows = 0;

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

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

        // Set the XML command type to insert.
cmd.XmlCommandType = OracleXmlCommandType.Insert;

    // Set the XML document.
    cmd.CommandText = "<?xml version="1.0"?>
    *<ROWSET>
    *  <MYROW num = "1">
    *     <EMPLOYEE_ID>1234</EMPLOYEE_ID>
    *     <LAST_NAME>Smith</LAST_NAME>
    *     <EMAIL>Smith@Oracle.com</EMAIL>
    *     <HIRE_DATE>1982-01-23T00:00:00.000</HIRE_DATE>
    *     <JOB_ID>IT_PROG</JOB_ID>
    *  </MYROW>
    *<MYROW num = "2">
    *     <EMPLOYEE_ID>1235</EMPLOYEE_ID>
    *     <LAST_NAME>Barney</LAST_NAME>
    *     <EMAIL>Barney@Oracle.com</EMAIL>
    *     <HIRE_DATE>1982-01-23T00:00:00.000</HIRE_DATE>
    *     <JOB_ID>IT_PROG</JOB_ID>
    *  </MYROW>
    </ROWSET>";

    // 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.UpdateColumnsList = UpdateColumnsList;
    cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
    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"?>
    *<ROWSET>
    *  <MYROW num = "1">
    *     <EMPLOYEE_ID>1234</EMPLOYEE_ID>
    *     <LAST_NAME>Adams</LAST_NAME>
    *  </MYROW>
    </ROWSET>";

    // 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;
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"?>
"<ROWSET>
"<MYROW num = '1'>
"<EMPLOYEE_ID>1234</EMPLOYEE_ID>
"</MYROW>
"<MYROW num = '2'>
"<EMPLOYEE_ID>1235</EMPLOYEE_ID>
"</MYROW>
"</ROWSET>";

// 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();
con.Close();
con.Dispose();
}
}

Requirements

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleXmlSaveProperties Members
- OracleXmlSaveProperties Constructor
- OracleXmlSaveProperties Properties
- OracleXmlSaveProperties Public Methods
OracleXmlSaveProperties Members

OracleXmlSaveProperties members are listed in the following tables:

OracleXmlSaveProperties Constructor
OracleXmlSaveProperties constructors are listed in Table 5–7.

Table 5–7 OracleXmlSaveProperties Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleXmlSaveProperties constructor</td>
<td>Instantiates a new instance of the OracleXmlSaveProperties class</td>
</tr>
</tbody>
</table>

OracleXmlSaveProperties Properties
The OracleXmlSaveProperties properties are listed in Table 5–8.

Table 5–8 OracleXmlSaveProperties Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyColumnsList</td>
<td>Specifies the list of columns used as a key to locate existing rows for update or delete using an XML document</td>
</tr>
<tr>
<td>RowTag</td>
<td>Specifies the value for the XML element that identifies a row of data in an XML document</td>
</tr>
<tr>
<td>Table</td>
<td>Specifies the name of the table or view to which changes are saved</td>
</tr>
<tr>
<td>UpdateColumnsList</td>
<td>Specifies the list of columns to update or insert</td>
</tr>
<tr>
<td>Xslt</td>
<td>Specifies the XSL document used for XML transformation using XSLT</td>
</tr>
<tr>
<td>XsltParams</td>
<td>Specifies the parameters for the XSLT document specified in the Xslt property</td>
</tr>
</tbody>
</table>

OracleXmlSaveProperties Public Methods
The OracleXmlSaveProperties public methods are listed in Table 5–9.

Table 5–9 OracleXmlSaveProperties Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleXmlSaveProperties object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members
OracleXmlSaveProperties Properties

The OracleXmlSaveProperties properties are listed in Table 5–10.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyColumnsList</td>
<td>Specifies the list of columns used as a key to locate existing rows for</td>
</tr>
<tr>
<td></td>
<td>update or delete using an XML document</td>
</tr>
<tr>
<td>RowTag</td>
<td>Specifies the value for the XML element that identifies a row of data</td>
</tr>
<tr>
<td></td>
<td>in an XML document</td>
</tr>
<tr>
<td>Table</td>
<td>Specifies the name of the table or view to which changes are saved</td>
</tr>
<tr>
<td>UpdateColumnsList</td>
<td>Specifies the list of columns to update or insert</td>
</tr>
<tr>
<td>Xslt</td>
<td>Specifies the XSL document used for XML transformation using XSLT</td>
</tr>
<tr>
<td>XsltParams</td>
<td>Specifies the parameters for the XSLT document specified in the Xslt</td>
</tr>
<tr>
<td></td>
<td>property</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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.
OracleXmlSaveProperties Properties

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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

```csharp
// 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" on page 1-1
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

Table

This property specifies the name of the table or view to which changes are saved.

Declaration

```csharp
// 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" on page 1-1
- 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" on page 1-1
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

**Xslt**

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.
**OracleXmlSaveProperties Properties**

**XsltParams**

This property specifies the parameters for the XSLT document specified in the `Xslt` property.

**Declaration**

```csharp
// 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" on page 1-1
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members
OracleXmlSaveProperties Public Methods

The OracleXmlSaveProperties public methods are listed in Table 5–11.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleXmlSaveProperties object</td>
</tr>
</tbody>
</table>

Clone

This method creates a copy of an OracleXmlSaveProperties object.

Declaration

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

Return Value

An OracleXmlSaveProperties object

Implements

ICloneable

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties 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 Oracle9i Release 2 (9.2) and later.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleXmlStream Members
- OracleXmlStream Constructor
- OracleXmlStream Static Methods
- OracleXmlStream Instance Properties
- OracleXmlStream Instance Methods
OracleXmlStream Class

OracleXmlStream Members

OracleXmlStream members are listed in the following tables:

**OracleXmlStream Constructors**
The OracleXmlStream constructors are listed in Table 5–12.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleXmlStream Constructor</td>
<td>Creates an instance of an OracleXmlStream object which provides a Stream representation of the XML data stored in an OracleXmlType</td>
</tr>
</tbody>
</table>

**OracleXmlStream Static Methods**
The OracleXmlStream static methods are listed in Table 5–13.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleXmlStream Instance Properties**
The OracleXmlStream instance properties are listed in Table 5–14.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether the XML stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether forward and backward seek operation can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Connection</td>
<td>Indicates the OracleConnection that is used to retrieve the XML data</td>
</tr>
<tr>
<td>Length</td>
<td>Indicates the number of bytes in the XML stream</td>
</tr>
<tr>
<td>Position</td>
<td>Gets or sets the byte position within the stream</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the XML data, starting from the first character in the stream as a string</td>
</tr>
</tbody>
</table>

**OracleXmlStream Instance Methods**
The OracleXmlStream instance methods are listed in Table 5–15.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleXmlStream object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current stream and releases any resources associated with it</td>
</tr>
</tbody>
</table>
### Table 5–15 (Cont.) OracleXmlStream Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Releases resources allocated by this object</td>
</tr>
<tr>
<td>EndRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Flush</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Flush</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a specified amount from the current stream instance and populates the array buffer (Overloaded)</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Seek</td>
<td>Sets the position within the current stream and returns the new position within the current stream</td>
</tr>
<tr>
<td>SetLength</td>
<td>Not Supported</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Write</td>
<td>Not Supported</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleXmlStream Class
- OracleXmlStream Members
OracleXmlStream Static Methods

The OracleXmlStream static methods are listed in Table 5–16.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleXmlStream Class
- OracleXmlStream Members
OracleXmlStream Instance Properties

The OracleXmlStream instance properties are listed in Table 5–17.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether the XML stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether forward and backward seek operation can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Connection</td>
<td>Indicates the OracleConnection that is used to retrieve the XML data</td>
</tr>
<tr>
<td>Length</td>
<td>Indicates the number of bytes in the XML stream</td>
</tr>
<tr>
<td>Position</td>
<td>Gets or sets the byte position within the stream</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the XML data, starting from the first character in the stream as a string</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleXmlStream Class
- OracleXmlStream Members

CanRead

Overrides Stream

This property indicates whether the XML stream can be read.

Declaration

```csharp
// 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" on page 1-4
- OracleXmlStream Class
- OracleXmlStream Members

CanSeek

Overrides Stream

This property indicates whether forward and backward seek operation can be performed.

Declaration

```csharp
// 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" on page 1-4
- 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" on page 1-4
- OracleXmlStream Class
- OracleXmlStream Members

**Length**
Overides 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.
Position

Overrides Stream
This property gets or sets the byte position within the stream.

Declaration

```csharp
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" on page 1-4
- OracleXmlStream Class
- OracleXmlStream Members

Value
This property returns the XML data, starting from the first character of the stream as a string.

Declaration

```csharp
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" on page 1-4
- OracleXmlStream Class
- OracleXmlStream Members
OracleXmlStream Instance Methods

The OracleXmlStream instance methods are listed in Table 5–18.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleXmlStream object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current stream and releases any resources associated with it</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated by this object</td>
</tr>
<tr>
<td>EndRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Flush</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a specified amount from the current XML stream instance and populates the array buffer (Overloaded)</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Seek</td>
<td>Sets the position within the current stream and returns the new position within the current stream</td>
</tr>
<tr>
<td>SetLength</td>
<td>Not Supported</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Write</td>
<td>Not Supported</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleXmlStream Class
- OracleXmlStream Members

Clone

This method creates a copy of an OracleXmlStream object.

Declaration

```csharp
// 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.

See Also:

● "Oracle.DataAccess.Types Namespace" on page 1-4
● 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" on page 1-4
● 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.
OracleXmlStream Class

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- 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 Oracle9i Release 2 (9.2) or higher.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 5–19.

**Table 5–19 OracleXmlType Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleXmlType Constructors</td>
<td>Creates an instance of the OracleXmlType class (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleXmlType Static Methods**
The OracleXmlType static methods are listed in Table 5–20.

**Table 5–20 OracleXmlType Static Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleXmlType Instance Properties**
The OracleXmlType instance properties are listed in Table 5–21.

**Table 5–21 OracleXmlType Instance Properties**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Indicates the OracleConnection that is used to retrieve and store XML data in the OracleXmlType</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Indicates whether or not the OracleXmlType is empty</td>
</tr>
<tr>
<td>IsFragment</td>
<td>Indicates whether the XML data is a collection of XML elements or a well-formed XML document</td>
</tr>
<tr>
<td>IsSchemaBased</td>
<td>Indicates whether or not the XML data represented by the OracleXmlType is based on an XML schema</td>
</tr>
<tr>
<td>RootElement</td>
<td>Represents the name of the top-level element of the schema-based XML data contained in the OracleXmlType</td>
</tr>
<tr>
<td>Schema</td>
<td>Represents the XML schema of the XML data contained in the OracleXmlType</td>
</tr>
<tr>
<td>SchemaUrl</td>
<td>Represents the database for the XML schema of the XML data contained in the OracleXmlType.</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the XML data starting from the first character in the current instance as a string</td>
</tr>
</tbody>
</table>

**OracleXmlType Instance Methods**
The OracleXmlType instance methods are listed in Table 5–22.

**Table 5–22 OracleXmlType Instance Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of the OracleXmlType instance</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases the resources allocated by this OracleXmlType object</td>
</tr>
</tbody>
</table>
### Table 5–22 (Cont.) OracleXmlType Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Extract</td>
<td>Extracts a subset from the XML data using the given XPath expression (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetStream</td>
<td>Returns an instance of OracleXmlStream which provides a read-only stream of the XML data stored in this OracleXmlType instance</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetXmlDocument</td>
<td>Returns a XmlDocument object containing the XML data stored in this OracleXmlType instance</td>
</tr>
<tr>
<td>GetXmlReader</td>
<td>Returns a XmlTextReader object that can be used to manipulate XML data directly using the .NET Framework classes and methods</td>
</tr>
<tr>
<td>IsExists</td>
<td>Checks for the existence of a particular set of nodes identified by the given XPath expression in the XML data (Overloaded)</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Transform</td>
<td>Transforms the OracleXmlType into another OracleXmlType instance using the given XSL document (Overloaded)</td>
</tr>
<tr>
<td>Update</td>
<td>Updates the XML node or fragment identified by the given XPath expression in the current OracleXmlType instance (Overloaded)</td>
</tr>
<tr>
<td>Validate</td>
<td>Validates whether the XML data in the OracleXmlType object conforms to the given XML schema.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.

**OracleXmlType(OracleClob)**

This constructor creates an instance of the OracleXmlType class using the XML data contained in an OracleClob object.

**Declaration**

```csharp
// C#
public OracleXmlType(OracleClob oraClob);
```

**Parameters**
- **oraClob**
  An OracleClob object.

**Exceptions**
- ArgumentException - 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.
OracleXmlType Constructors

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" on page 1-4
- 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.
OracleXmlType Class

- 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" on page 1-4
- 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" on page 1-4
- OracleXmlType Class
- OracleXmlType Members
OracleXmlType Static Methods

The OracleXmlType static methods are listed in Table 5–23.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleXmlType Class
- OracleXmlType Members
OracleXmlType Class

OracleXmlType Instance Properties

The OracleXmlType instance properties are listed in Table 5–24.

Table 5–24  OracleXmlType Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Indicates the OracleConnection that is used to retrieve and store XML data in the OracleXmlType</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Indicates whether or not the OracleXmlType is empty</td>
</tr>
<tr>
<td>IsFragment</td>
<td>Indicates whether the XML data is a collection of XML elements or a well-formed XML document</td>
</tr>
<tr>
<td>IsSchemaBased</td>
<td>Indicates whether or not the XML data represented by the OracleXmlType is based on an XML schema</td>
</tr>
<tr>
<td>RootElement</td>
<td>Represents the name of the top-level element of the schema-based XML data contained in the OracleXmlType</td>
</tr>
<tr>
<td>Schema</td>
<td>Represents the XML schema of the XML data contained in the OracleXmlType</td>
</tr>
<tr>
<td>SchemaUrl</td>
<td>Represents URL in the database for the XML schema of the XML data contained in the OracleXmlType</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the XML data starting from the first character in the current instance as a string</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleXmlType Class
- OracleXmlType Members

Connection

This property indicates the OracleConnection that is used to retrieve and store XML data in the OracleXmlType.

Declaration

```csharp
// 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.
### 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.

### 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleXmlType Class
- OracleXmlType Members

Schema

This property represents the XML schema for the XML data contained in the OracleXmlType.
OracleXmlType Instance Properties

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" on page 1-4
- OracleXmlType Class
- OracleXmlType Members

SchemaUrl

This property represents the XML schema 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" on page 1-4
- 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" on page 1-4
- OracleXmlType Class
- OracleXmlType Members
OracleXmlType Instance Methods

The OracleXmlType instance methods are listed in Table 5–25.

Table 5–25 OracleXmlType Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of the OracleXmlType instance</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases the resources allocated by this OracleXmlType object</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Extract</td>
<td>Extracts a subset from the XML data using the given XPath expression (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetStream</td>
<td>Returns an instance of OracleXmlStream which provides a read-only stream of the XML data stored in this OracleXmlType instance</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetXmlDocument</td>
<td>Returns a XmlDocument object containing the XML data stored in this OracleXmlType instance</td>
</tr>
<tr>
<td>GetXmlReader</td>
<td>Returns a XmlTextReader object that can be used to manipulate XML data directly using the .NET Framework classes and methods</td>
</tr>
<tr>
<td>IsExists</td>
<td>Checks for the existence of a particular set of nodes identified by the given XPath expression in the XML data (Overloaded)</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Transform</td>
<td>Transforms the OracleXmlType into another OracleXmlType instance using the given XSL document (Overloaded)</td>
</tr>
<tr>
<td>Update</td>
<td>Updates the XML node or fragment identified by the given XPath expression in the current OracleXmlType instance (Overloaded)</td>
</tr>
<tr>
<td>Validate</td>
<td>Validates whether the XML data in the OracleXmlType object conforms to the given XML schema.</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleXmlType Class
- OracleXmlType Members

Clone

This method creates a copy of this OracleXmlType instance.

Declaration

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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:


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" on page 1-4
- 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 for namespace resolution.
OracleXmlType Class

- 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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- OracleXmlType Class
- OracleXmlType Members

**GetXmlDocument**

This public method returns a XmlDocument object containing the XML data stored in this OracleXmlType instance.

**Declaration**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- `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.
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.
- ArgumentException - The xpathExpr is null or zero-length.
- InvalidOperation - 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" on page 1-4
- 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" on page 1-4
  - `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

```csharp
// 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" on page 1-4
- `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

```csharp
// 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" on page 1-4
- `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" on page 1-4
- 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:
  ```xml
  ```

- `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" on page 1-4
- 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" on page 1-4

■ 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
  See Also:

■ OracleXmlType Members
The string parameter used for namespace resolution of the XPath expression. 
\textit{nsMap} has zero or more namespaces separated by spaces. \textit{nsMap} can be null.

- \textbf{value}
  The new value as an \textit{OracleXmlType} object.

\textbf{Exceptions}

- \textit{ObjectDisposedException} - The object is already disposed.
- \textit{ArgumentNullException} - The \textit{xpathExpr} is null or zero-length.
- \textit{InvalidOperationException} - The \textit{OracleConnection} is not open or has been closed during the lifetime of the object.

\textbf{Remarks}

The default namespace is ignored if its value is an empty string.

\textbf{See Also:}

- "Oracle.DataAccess.Types Namespace" on page 1-4
- \textit{OracleXmlType} Class
- \textit{OracleXmlType Members}

\textbf{Update(string, XmlNameSpaceManager, OracleXmlType)}

This method updates the XML nodes identified by the given XPath expression with the XML data stored in the given \textit{OracleXmlType} value and a .NET \textit{XmlNameSpaceManager} object for namespace resolution.

\textbf{Declaration}

// C#
public void Update(string \textit{xpathExpr}, XmlNameSpaceManager \textit{nsMgr}, OracleXmlType \textit{value});

\textbf{Parameters}

- \textit{xpathExpr}
  The XPath expression that identifies the nodes to update.
- \textit{nsMgr}
  The .NET \textit{XmlNameSpaceManager} object used for namespace resolution of the XPath expression. \textit{nsMgr} can be null.
- \textit{value}
  The new value as an \textit{OracleXmlType} object.

\textbf{Exceptions}

- \textit{ObjectDisposedException} - The object is already disposed.
- \textit{ArgumentNullException} - The \textit{xpathExpr} is null or zero-length.
- \textit{InvalidOperationException} - The \textit{OracleConnection} is not open or has been closed during the lifetime of the object.

\textbf{Remarks}

The default namespace is ignored if its value is an empty string.
**Validate**

This method validates whether the XML data in the `OracleXmlType` object conforms to the given XML schema.

**Declaration**

```csharp
// 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.
This chapter describes the Oracle Data Provider for .NET globalization classes.

This chapter contains these topics:

- OracleGlobalization Class
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#

using System;
using Oracle.DataAccess.Client;

class OracleGlobalizationSample
{
    static void Main()
    {
        // Get thread's globalization info
        OracleGlobalization glob = OracleGlobalization.GetThreadInfo();

        // Prints "glob.Language = AMERICAN"
        Console.WriteLine("glob.Language = " + glob.Language);

        // Set language on thread's globalization info
        glob.Language = "FRENCH";
        OracleGlobalization.SetThreadInfo(glob);
        OracleGlobalization.GetThreadInfo(glob);

        // Prints "glob.Language = FRENCH"
        Console.WriteLine("glob.Language = " + glob.Language);

        glob.Dispose();
    }
}
Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleGlobalization Members
- OracleGlobalization Static Methods
- OracleGlobalization Properties
- OracleGlobalization Public Methods
- Oracle Database SQL Reference
- Oracle Database Globalization Support Guide
OracleGlobalization Members

OracleGlobalization members are listed in the following tables:

OracleGlobalization Static Methods
The OracleGlobalization static methods are listed in Table 6–1.

Table 6–1 OracleGlobalization Static Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetClientInfo</td>
<td>Returns an OracleGlobalization object that represents the Oracle globalization settings of the local computer (Overloaded)</td>
</tr>
<tr>
<td>GetThreadInfo</td>
<td>Returns or refreshes an OracleGlobalization instance that represents Oracle globalization settings of the current thread (Overloaded)</td>
</tr>
<tr>
<td>SetThreadInfo</td>
<td>Sets Oracle globalization parameters to the current thread</td>
</tr>
</tbody>
</table>

OracleGlobalization Properties
The OracleGlobalization properties are listed in Table 6–2.

Table 6–2 OracleGlobalization Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>Specifies the calendar system</td>
</tr>
<tr>
<td>ClientCharacterSet</td>
<td>Specifies a client character set</td>
</tr>
<tr>
<td>Comparison</td>
<td>Specifies a method of comparison for WHERE clauses and comparison in PL/SQL blocks</td>
</tr>
<tr>
<td>Currency</td>
<td>Specifies the string to use as a local currency symbol for the L number format element</td>
</tr>
<tr>
<td>DateFormat</td>
<td>Specifies the date format for Oracle Date type as a string</td>
</tr>
<tr>
<td>DateLanguage</td>
<td>Specifies the language used to spell day and month names and date abbreviations</td>
</tr>
<tr>
<td>DualCurrency</td>
<td>Specifies the dual currency symbol, such as Euro, for the U number format element</td>
</tr>
<tr>
<td>ISOCurrency</td>
<td>Specifies the string to use as an international currency symbol for the C number format element</td>
</tr>
<tr>
<td>Language</td>
<td>Specifies the default language of the database</td>
</tr>
<tr>
<td>LengthSemantics</td>
<td>Enables creation of CHAR and VARCHAR2 columns using either byte or character (default) length semantics</td>
</tr>
<tr>
<td>NCharConversionException</td>
<td>Determines whether data loss during an implicit or explicit character type conversion reports an error</td>
</tr>
<tr>
<td>NumericCharacters</td>
<td>Specifies the characters used for the decimal character and the group separator character for numeric values in strings</td>
</tr>
<tr>
<td>Sort</td>
<td>Specifies the collating sequence for ORDER by clause</td>
</tr>
<tr>
<td>Territory</td>
<td>Specifies the name of the territory</td>
</tr>
<tr>
<td>TimeStampFormat</td>
<td>Specifies the string format for Timestamp types</td>
</tr>
<tr>
<td>TimeStampTZFormat</td>
<td>Specifies the string format for TimestampTZ types</td>
</tr>
<tr>
<td>TimeZone</td>
<td>Specifies the time zone region name</td>
</tr>
</tbody>
</table>
**OracleGlobalization Public Methods**

OracleGlobalization public methods are listed in Table 6–6.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleGlobalization object</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members
OracleGlobalization Static Methods

The OracleGlobalization static methods are listed in Table 6–4.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetClientInfo</td>
<td>Returns an OracleGlobalization object that represents the Oracle globalization settings of the local computer (Overloaded)</td>
</tr>
<tr>
<td>GetThreadInfo</td>
<td>Returns or refreshes an OracleGlobalization instance that represents Oracle globalization settings of the current thread (Overloaded)</td>
</tr>
<tr>
<td>SetThreadInfo</td>
<td>Sets Oracle globalization parameters to the current thread</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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#

using System;
using Oracle.DataAccess.Client;

class GetClientInfoSample
{
    static void Main()
    {
        // Get client's globalization info
        OracleGlobalization glob = OracleGlobalization.GetClientInfo();

        // Prints 'glob.Language = AMERICAN'
        Console.WriteLine("glob.Language = " + glob.Language);

        glob.Dispose();
    }
}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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#

using System;
using Oracle.DataAccess.Client;

class GetClientInfoSample
{
    static void Main()
    {
        // Get client's globalization info
        OracleGlobalization glob = OracleGlobalization.GetClientInfo();

        // Prints 'glob.Language = AMERICAN'
        Console.WriteLine("glob.Language = " + glob.Language);

        // Get client's globalization info using overload
        OracleGlobalization.GetClientInfo(glob);
    }
}
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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

GetThreadInfo()

This method returns an OracleGlobalization instance of the current thread.

Declaration

```csharp
// 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

```csharp
// C#

using System;
using Oracle.DataAccess.Client;
```
```csharp
class GetThreadInfoSample
{
    static void Main()
    {
        // Get thread's globalization info
        OracleGlobalization glob = OracleGlobalization.GetThreadInfo();

        // Prints 'glob.Language = AMERICAN'
        Console.WriteLine("glob.Language = " + glob.Language);

        // Get thread's globalization info using overloaded
        OracleGlobalization.GetThreadInfo(glob);

        // Prints 'glob.Language = AMERICAN'
        Console.WriteLine("glob.Language = " + glob.Language);

        glob.Dispose();
    }
}
```

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

**GetThreadInfo(OracleGlobalization)**

This method refreshes the `OracleGlobalization` object with the globalization settings of the current thread.

**Declaration**

```csharp
// 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**

```csharp
// C#
using System;
using Oracle.DataAccess.Client;

class GetThreadInfoSample
{
    static void Main()
    {
        // Get thread's globalization info
```
OracleGlobalization glob = OracleGlobalization.GetThreadInfo();

// Prints "glob.Language = AMERICAN"
Console.WriteLine('glob.Language = ' + glob.Language);

// Get thread's globalization info using overloaded
OracleGlobalization.GetThreadInfo(glob);

// Prints "glob.Language = AMERICAN"
Console.WriteLine('glob.Language = ' + glob.Language);

glob.Dispose();
}
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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#

using System;
using Oracle.DataAccess.Client;

class SetThreadInfoSample
{
  static void Main()
  {
    // Get thread's globalization info
    OracleGlobalization glob1 = OracleGlobalization.GetThreadInfo();

    // Prints "glob1.Language = AMERICAN"
    Console.WriteLine('glob1.Language = ' + glob1.Language);
  }
}
// Set language on thread's globalization info
glob1.Language = "FRENCH";
OracleGlobalization.SetThreadInfo(glob1);
OracleGlobalization glob2 = OracleGlobalization.GetThreadInfo();

// Prints "glob2.Language = FRENCH"
Console.WriteLine("glob2.Language = " + glob2.Language);

glob1.Dispose();
glob2.Dispose();

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members
OracleGlobalization Properties

The OracleGlobalization properties are listed in Table 6–5.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>Specifies the calendar system</td>
</tr>
<tr>
<td>ClientCharacterSet</td>
<td>Specifies a client character set</td>
</tr>
<tr>
<td>Comparison</td>
<td>Specifies a method of comparison for WHERE clauses and comparison in PL/SQL blocks</td>
</tr>
<tr>
<td>Currency</td>
<td>Specifies the string to use as a local currency symbol for the L number format element</td>
</tr>
<tr>
<td>DateFormat</td>
<td>Specifies the date format for Oracle Date type as a string</td>
</tr>
<tr>
<td>DateLanguage</td>
<td>Specifies the language used to spell day and month names and date abbreviations</td>
</tr>
<tr>
<td>DualCurrency</td>
<td>Specifies the dual currency symbol, such as Euro, for the U number format element</td>
</tr>
<tr>
<td>ISOICurrency</td>
<td>Specifies the string to use as an international currency symbol for the C number format element</td>
</tr>
<tr>
<td>Language</td>
<td>Specifies the default language of the database</td>
</tr>
<tr>
<td>LengthSemantics</td>
<td>Enables creation of CHAR and VARCHAR2 columns using either byte or character (default) length semantics</td>
</tr>
<tr>
<td>NCharConversionException</td>
<td>Determines whether data loss during an implicit or explicit character type conversion reports an error</td>
</tr>
<tr>
<td>NumericCharacters</td>
<td>Specifies the characters used for the decimal character and the group separator character for numeric values in strings</td>
</tr>
<tr>
<td>Sort</td>
<td>Specifies the collating sequence for ORDER by clause</td>
</tr>
<tr>
<td>Territory</td>
<td>Specifies the name of the territory</td>
</tr>
<tr>
<td>TimeStampFormat</td>
<td>Specifies the string format for TimeStamp types</td>
</tr>
<tr>
<td>TimeStampTZFormat</td>
<td>Specifies the string format for TimeStampTZ types</td>
</tr>
<tr>
<td>TimeZone</td>
<td>Specifies the time zone region name</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

Calendar

This property specifies the calendar system.

Declaration

```csharp
// 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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

ClientCharacterSet
This property specifies a client character set.

Declaration
// C#
public string ClientCharacterSet {get;}

Property Value
A string that 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" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

Date Format
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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members
- Oracle Database SQL Reference for further information on the L number format element
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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" on page 1-1
- 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" on page 1-1
- 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 ISO_currency {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" on page 1-1
- 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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

LengthSemantics

This property indicates whether CHAR and VARCHAR2 columns use byte or character (default) length semantics.

Declaration

```csharp
// 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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

NCharConversionException

This property determines whether data loss during an implicit or explicit character type conversion reports an error.

Declaration

```csharp
// 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 of NLS_NCHAR_CONV_EXCP is False, unless it is overridden by a setting in the INIT.ORA file.
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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**

```csharp
// 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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

### Sort

This property specifies the collating sequence for `ORDER` by clause.

**Declaration**

```csharp
// 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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members
**Territory**

This property specifies the name of the territory.

**Declaration**

```csharp
// 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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members
- Oracle Database Globalization Support Guide.

---

**Time Stamp Format**

This property specifies the string format for TimeStamp types.

**Declaration**

```csharp
// 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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

---

**Time Stamp TZ Format**

This property specifies the string format for TimeStampTZ types.
OracleGlobalization Properties

Declaration

```csharp
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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

TimeZone

This property specifies the time zone region name or hour offset.

Declaration

```csharp
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" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members
OracleGlobalization public methods are listed in Table 6–6.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleGlobalization object</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from Component</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members

Clone

This method creates a copy of an OracleGlobalization object.

Declaration

```csharp
// 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.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleGlobalization Class
- OracleGlobalization Members
This chapter describes the Oracle Data Provider for .NET failover classes and enumerations.

This chapter contains these topics:

- **OracleFailoverEventArgs Class**
- **OracleFailoverEventHandler Delegate**
- **FailoverEvent Enumeration**
- **FailoverReturnCode Enumeration**
- **FailoverType Enumeration**
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
// Transparent Application Failover (TAF) Setup
// Refer Oracle® Database Net Services Administrator's Guide

using System;
using System.Threading;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleFailoverEventArgsSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Register the event handler OnFailover
        con.Failover += new OracleFailoverEventHandler(OnFailover);

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

    // TAF callback function
    static FailoverReturnCode OnFailover(object sender,
        OracleFailoverEventArgs eventArgs)
    {
        switch (eventArgs.FailoverEvent)
        {
        case FailoverEvent.Begin:
            
        
    
}
Console.WriteLine("FailoverEvent.Begin - Failover is starting");
Console.WriteLine("FailoverType = " + eventArgs.FailoverType);
break;
}
case FailoverEvent.End:
{
    Console.WriteLine("FailoverEvent.End - Failover was successful");
    break;
}
case FailoverEvent.Reauth:
{
    Console.WriteLine("FailoverEvent.Reauth - User reauthenticated");
    break;
}
case FailoverEvent.Error:
{
    Console.WriteLine("FailoverEvent.Error - Failover was unsuccessful");

    // Sleep for 3 sec and Retry
    Thread.Sleep(3000);
    return FailoverReturnCode.Retry;
}
case FailoverEvent.Abort:
{
    Console.WriteLine("FailoverEvent.Abort - Failover was unsuccessful");
    break;
}
default:
{
    Console.WriteLine("Invalid FailoverEvent : " + eventArgs.FailoverEvent);
    break;
}
}
return FailoverReturnCode.Success;

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- 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 7–1.

Table 7–1 OracleFailoverEventArgs Static Methods
<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleFailoverEventArgs Properties
The OracleFailoverEventArgs properties are listed in Table 7–2.

Table 7–2 OracleFailoverEventArgs Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FailoverType</td>
<td>Specifies the type of failover the client has requested</td>
</tr>
<tr>
<td>FailoverEvent</td>
<td>Indicates the state of the failover</td>
</tr>
</tbody>
</table>

OracleFailoverEventArgs Public Methods
The OracleFailoverEventArgs public methods are listed in Table 7–3.

Table 7–3 OracleFailoverEventArgs Public Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleFailoverEventArgs Class
- "FailoverType Enumeration" on page 7-12
OracleFailoverEventArgs Static Methods

The OracleFailoverEventArgs static methods are listed in Table 7–1.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members
OracleFailoverEventArgs Properties

The `OracleFailoverEventArgs` properties are listed in Table 7–5.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FailoverType</td>
<td>Specifies the type of failover the client has requested</td>
</tr>
<tr>
<td>FailoverEvent</td>
<td>Indicates the state of the failover</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- `OracleFailoverEventArgs` Class
- `OracleFailoverEventArgs` Members

**FailoverType**

This property indicates the state of the failover.

**Declaration**

```csharp
// C#
public FailoverType FailoverType {get;}
```

**Property Value**

A `FailoverType` enumeration value.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- `OracleFailoverEventArgs` Class
- `OracleFailoverEventArgs` Members
- "FailoverType Enumeration" on page 7-12

**FailoverEvent**

This property indicates the state of the failover.

**Declaration**

```csharp
// C#
public FailoverEvent FailoverEvent {get;}
```

**Property Value**

A `FailoverEvent` enumerated value.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- `OracleFailoverEventArgs` Class
- `OracleFailoverEventArgs` Members
- "FailoverEvent Enumeration" on page 7-10
OracleFailoverEventArgs Public Methods

The OracleFailoverEventArgs public methods are listed in Table 7-6.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members
OracleFailoverEventHandler Delegate

The `OracleFailoverEventHandler` represents the signature of the method that handles the `OracleConnection.Failover` event.

**Declaration**

```csharp
// 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:

```csharp
ConObj.Failover += new OracleFailoverEventHandler(OnFailover);
```

The definition of the callback function `OnFailover` can be as follows:

```csharp
public FailoverReturnCode OnFailover(object sender, OracleFailoverEventArgs eventArgs)
```

**Example**

```csharp
// Transparent Application Failover (TAF) Setup
// Refer Oracle® Database Net Services Administrator's Guide

using System;
using System.Threading;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleFailoverEventHandlerSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        // Register the event handler OnFailover
        con.Failover += new OracleFailoverEventHandler(OnFailover);

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

    public FailoverReturnCode OnFailover(object sender, OracleFailoverEventArgs eventArgs)
    {
        // Implement the failover logic here...
        return FailoverReturnCode.Success;
    }
}
```
// TAF callback function
static FailoverReturnCode OnFailover(object sender,
   OracleFailoverEventArgs eventArgs)
{
   switch (eventArgs.FailoverEvent)
   {
   case FailoverEvent.Begin:
   {
      Console.WriteLine("FailoverEvent.Begin - Failover is starting");
      Console.WriteLine("FailoverType = " + eventArgs.FailoverType);
      break;
   }
   case FailoverEvent.End:
   {
      Console.WriteLine("FailoverEvent.End - Failover was successful");
      break;
   }
   case FailoverEvent.Reauth:
   {
      Console.WriteLine("FailoverEvent.Reauth - User reauthenticated");
      break;
   }
   case FailoverEvent.Error:
   {
      Console.WriteLine("FailoverEvent.Error - Failover was unsuccessful");

      // Sleep for 3 sec and Retry
      Thread.Sleep(3000);
      return FailoverReturnCode.Retry;
   }
   case FailoverEvent.Abort:
   {
      Console.WriteLine("FailoverEvent.Abort - Failover was unsuccessful");
      break;
   }
   default:
   {
      Console.WriteLine("Invalid FailoverEvent : " + eventArgs.FailoverEvent);
      break;
   }
   }
   return FailoverReturnCode.Success;
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-1
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members
FailoverEvent Enumeration

FailoverEvent enumerated values are used to explicitly specify the state of the failover.

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

<table>
<thead>
<tr>
<th>Member Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FailoverEvent.Begin</td>
<td>Indicates that failover has detected a lost connection and that failover is starting.</td>
</tr>
<tr>
<td>FailoverEvent.End</td>
<td>Indicates successful completion of failover.</td>
</tr>
<tr>
<td>FailoverEvent.Abort</td>
<td>Indicates that failover was unsuccessful, and there is no option of retrying.</td>
</tr>
<tr>
<td>FailoverEvent.Error</td>
<td>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.</td>
</tr>
<tr>
<td>FailoverEvent.Reauth</td>
<td>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.</td>
</tr>
</tbody>
</table>

See Also:
- FailoverEvent Enumeration on page 7-10
- "OracleFailoverEventArgs Class" on page 7-2
- "FailoverEvent" on page 7-6
- 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 7–8 lists the FailoverReturnCode enumeration values with a description of each enumerated value.

Table 7–8   FailoverReturnCode Enumeration Values

<table>
<thead>
<tr>
<th>Member Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FailoverReturnCode.Retry</td>
<td>Requests ODP.NET to retry failover in case FailoverEvent.Error is passed to the application</td>
</tr>
<tr>
<td>FailoverReturnCode.Success</td>
<td>Requests ODP.NET to proceed so that the application receive more notifications, if any</td>
</tr>
</tbody>
</table>

See Also:

- FailoverEvent Enumeration on page 7-10
- "OracleFailoverEventArgs Class" on page 7-2
- "FailoverEvent" on page 7-6
- 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 7–9 lists all the FailoverType enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FailoverType.Session</td>
<td>Indicates that the user has requested only session failover.</td>
</tr>
<tr>
<td>FailoverType.Select</td>
<td>Indicates that the user has requested select and session failover.</td>
</tr>
</tbody>
</table>

See Also:
- FailoverEvent Enumeration on page 7-10
- "OracleFailoverEventArgs Class" on page 7-2
- "FailoverType" on page 7-6
- Oracle Real Application Clusters Quick Start
- Oracle Net Services Reference Guide
This chapter describes the large object and REF CURSOR objects provided by Oracle Data Provider for .NET.

This chapter contains these topics:

- ODP.NET Types (ODP.NET LOB objects) consisting of these object classes:
  - OracleBFile Class
  - OracleBlob Class
  - OracleClob Class
  - OracleRefCursor Class

All offsets are 0-based for all ODP.NET LOB 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**

// Database Setup
/*
-- User must have CREATE ANY DIRECTORY privilege
CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';
/
*/

// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleBFileSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated above and create a file
        // MyFile.txt with the text ABCDABC under MYDIR.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();
OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

// Open the OracleBFile
bFile.OpenFile();

// Read 7 bytes into readBuffer, starting at buffer offset 0
byte[] readBuffer = new byte[7];
int bytesRead = bFile.Read(readBuffer, 0, 7);

// Prints 'bytesRead = 7'
Console.WriteLine("bytesRead = " + bytesRead);

// Prints 'readBuffer = 65666768656667'
Console.Write("readBuffer = ");
for(int index = 0; index < readBuffer.Length; index++)
{
    Console.Write(readBuffer[index]);
}
Console.WriteLine();

// Search for the 2nd occurrence of a byte pattern {66,67}
// starting from byte offset 1 in the OracleBFile
long posFound = bFile.Search(pattern, 1, 2);

// Prints 'posFound = 6'
Console.WriteLine("posFound = " + posFound);

// Close the OracleBFile
bFile.CloseFile();
bFile.Close();
bFile.Dispose();
con.Close();
con.Dispose();
}

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 8–1.

Table 8–1 OracleBFile Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleBFile Constructors</td>
<td>Creates an instance of the OracleBFile class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleBFile Static Fields
OracleBFile static fields are listed in Table 8–2.

Table 8–2 OracleBFile Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSize</td>
<td>The static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 (2^32 - 1) bytes</td>
</tr>
</tbody>
</table>

OracleBFile Static Methods
OracleBFile static methods are listed in Table 8–3.

Table 8–3 OracleBFile Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleBFile Instance Properties
OracleBFile instance properties are listed in Table 8–4.

Table 8–4 OracleBFile Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether forward and backward seek operations can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether the LOB object supports writing</td>
</tr>
<tr>
<td>Connection</td>
<td>Indicates the connection used to read from a BFILE</td>
</tr>
<tr>
<td>DirectoryName</td>
<td>Indicates the directory alias of the BFILE</td>
</tr>
<tr>
<td>FileExists</td>
<td>Indicates whether or not the specified BFILE exists</td>
</tr>
<tr>
<td>FileName</td>
<td>Indicates the name of the BFILE</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Indicates whether the BFILE is empty or not</td>
</tr>
<tr>
<td>IsOpen</td>
<td>Indicates whether the BFILE has been opened by this instance or not</td>
</tr>
<tr>
<td>Length</td>
<td>Indicates the size of the BFILE data in bytes</td>
</tr>
</tbody>
</table>
### OracleBFile Instance Methods

OracleBFile instance methods are listed in Table 8–5.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleBFile object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current stream and releases any resources associated with the stream</td>
</tr>
<tr>
<td>CloseFile</td>
<td>Closes the BFILE referenced by the current BFILE instance</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares data referenced by the two OracleBFiles</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies data as specified (Overloaded)</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated by this object</td>
</tr>
<tr>
<td>EndRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares the LOB references</td>
</tr>
<tr>
<td>OpenFile</td>
<td>Opens the BFILE specified by the FileName and DirectoryName</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a specified amount of bytes from the OracleBFile instance and populates the buffer</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Search</td>
<td>Searches for a binary pattern in the current instance of an OracleBFile</td>
</tr>
<tr>
<td>Seek</td>
<td>Sets the position on the current LOB stream</td>
</tr>
<tr>
<td>SetLength</td>
<td>Not Supported</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

### Table 8–5  OracleBFile Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Indicates the current read position in the LOB stream</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the data, starting from the first byte in BFILE, as a byte array</td>
</tr>
</tbody>
</table>
### Table 8–5  (Cont.) OracleBFile Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>Not Supported</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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” on page 1-4
  - OracleBFile Class
  - OracleBFile Members

**OracleBFile(OracleConnection)**

This constructor creates an instance of the OracleBFile class with an OracleConnection object.

**Declaration**

```csharp
// 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” on page 1-4
  - 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" on page 1-4
- OracleBFile Class
- OracleBFile Members
OracleBFile Static Fields

OracleBFile static fields are listed in Table 8–6.

Table 8–6  OracleBFile Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSize</td>
<td>The static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 (2^32 - 1) bytes</td>
</tr>
</tbody>
</table>

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" on page 1-4
- OracleBFile Class
- OracleBFile Members
OracleBFile Static Methods

OracleBFile static methods are listed in Table 8–7.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBFile Class
- OracleBFile Members
OracleBFile Instance Properties

OracleBFile instance properties are listed in Table 8–8.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether forward and backward seek operations can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether the LOB object supports writing</td>
</tr>
<tr>
<td>Connection</td>
<td>Indicates the connection used to read from a BFILE</td>
</tr>
<tr>
<td>DirectoryName</td>
<td>Indicates the directory alias of the BFILE</td>
</tr>
<tr>
<td>FileExists</td>
<td>Indicates whether or not the specified BFILE exists</td>
</tr>
<tr>
<td>FileName</td>
<td>Indicates the name of the BFILE</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Indicates whether the BFILE is empty or not</td>
</tr>
<tr>
<td>IsOpen</td>
<td>Indicates whether the BFILE has been opened by this instance or not</td>
</tr>
<tr>
<td>Length</td>
<td>Indicates the size of the BFILE data in bytes</td>
</tr>
<tr>
<td>Position</td>
<td>Indicates the current read position in the LOB stream</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the data, starting from the first byte in BFILE, as a byte array</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBFile Class
- OracleBFile Members

CanRead

Overrides Stream

This instance property indicates whether the LOB stream can be read.

Declaration

```csharp
// 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" on page 1-4
- OracleBFile Class
- OracleBFile Members

CanSeek

Overrides Stream
This instance property indicates whether forward and backward seek operations can be performed.

**Declaration**

```csharp
// C#
public override bool CanSeek{get;}
```

**Property Value**

If forward and backward seek operations can be performed, returns `true`; otherwise, returns `false`.

**Remarks**

BFILE is read-only, therefore, the boolean value is always `false`.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBFile Class
- OracleBFile Members

---

**CanWrite**

Overides Stream

This instance property indicates whether the LOB object supports writing.

**Declaration**

```csharp
// 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" on page 1-4
- OracleBFile Class
- OracleBFile Members

---

**Connection**

This instance property indicates the connection used to read from a BFILE.

**Declaration**

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

**Property Value**

An object of OracleConnection.

**Exceptions**

ObjectDisposedException - The object is already disposed.
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" on page 1-4
- 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" on page 1-4
- OracleBFile Class
- OracleBFile Members
OracleBFile Instance Properties

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleBFile Class
- OracleBFile Members

Length

Overrrides 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" on page 1-4
- OracleBFile Class
- OracleBFile Members

Position

Overrrides 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.
OracleBFile Instance Properties

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" on page 1-4
- 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" on page 1-4
- OracleBFile Class
- OracleBFile Members
OracleBFile Instance Methods

OracleBFile instance methods are listed in Table 8–9.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleBFile object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current stream and releases any resources associated with the stream</td>
</tr>
<tr>
<td>CloseFile</td>
<td>Closes the BFILE referenced by the current BFILE instance</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares data referenced by the two OracleBFiles</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies data as specified (Overloaded)</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated by this object</td>
</tr>
<tr>
<td>EndRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares the LOB references</td>
</tr>
<tr>
<td>OpenFile</td>
<td>Opens the BFILE specified by the FileName and DirectoryName</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a specified amount of bytes from the OracleBFile instance and populates the buffer</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Search</td>
<td>Searches for a binary pattern in the current instance of an OracleBFile</td>
</tr>
<tr>
<td>Seek</td>
<td>Sets the position on the current LOB stream</td>
</tr>
<tr>
<td>SetLength</td>
<td>Not Supported</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Write</td>
<td>Not Supported</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>
Clone

This instance method creates a copy of an OracleBFile object.

**Declaration**

```csharp
// 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**

```csharp
// Database Setup
/*
-- User must have CREATE ANY DIRECTORY privilege
CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';
*/

// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CloneSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated above and create a file
        // MyFile.txt with the text ABCDABC under MYDIR.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile1 = new OracleBFile(con, "MYDIR", "MyFile.txt");
    }
}
```
// Prints "bFile1.Position = 0"
Console.WriteLine("bFile1.Position = " + bFile1.Position);

// Set the Position before calling Clone()
bFile1.Position = 1;

// Clone the OracleBFile
OracleBFile bFile2 = (OracleBFile) bFile1.Clone();

// Prints "bFile2.Position = 1"
Console.WriteLine("bFile2.Position = " + bFile2.Position);

bFile1.Close();
bFile1.Dispose();

bFile2.Close();
bFile2.Dispose();

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

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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 BFILES 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
// Database Setup
/*
-- User must have CREATE ANY DIRECTORY privilege
CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';
/ */

// C#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CompareSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated above and create a file
        // MyFile.txt with the text ABCDABC under MYDIR.
        // Note that the byte representation of the ABCDABC is 65666768656667
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile1 = new OracleBFile(con, "MYDIR", "MyFile.txt");
        OracleBFile bFile2 = new OracleBFile(con, "MYDIR", "MyFile.txt");

        // Open the OracleBFiles
        bFile1.OpenFile();
        bFile2.OpenFile();

        // Compare 2 bytes from the 1st byte of bFile1 and
        // the 5th byte of bFile2 onwards
        int result = bFile1.Compare(1, bFile2, 5, 2);

        // Prints 'result = 0' (Indicates the data is identical)
        Console.WriteLine("result = " + result);

        // Close the OracleBFiles
        bFile1.CloseFile();
        bFile2.CloseFile();
        bFile1.Close();
        bFile1.Dispose();
        bFile2.Close();
        bFile2.Dispose();
        con.Close();
        con.Dispose();
    }
}
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- `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**

```csharp
// 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" on page 1-4
- 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.
- InvalidOperation - 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" on page 1-4
- 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**

```csharp
// 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**

- `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" on page 1-4
- `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" on page 1-4

■ 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.

**IsEqual**

This instance method compares the LOB references.

**Declaration**

```csharp
// 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” on page 1-4
- `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" on page 1-4
- 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
// Database Setup
/*
-- User must have CREATE ANY DIRECTORY privilege
CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';
/*
*/

// C#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class ReadSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated above and create a file
        // MyFile.txt with the text ABCDABC under MYDIR.
        // Note that the byte representation of the ABCDABC is 656676865667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

        // Open the OracleBFile
        bFile.OpenFile();

        // Read 7 bytes into readBuffer, starting at buffer offset 0
        byte[] readBuffer = new byte[7];
        int bytesRead = bFile.Read(readBuffer, 0, 7);

        // Prints "bytesRead  = 7"
        Console.WriteLine("bytesRead  = " + bytesRead);

        // Prints "readBuffer = 656676865667"
        Console.Write("readBuffer = ");
        for(int index = 0; index < readBuffer.Length; index++)
        {
            Console.Write(readBuffer[index]);
        }
        Console.WriteLine();

        // Close the OracleBFile
        bFile.CloseFile();
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

// Database Setup
/*
-- User must have CREATE ANY DIRECTORY privilege
CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';
/
*/

// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class SearchSample
{
    static void Main()
    {
        // Create MYDIR directory object as indicated above and create a file
        // MyFile.txt with the text ABCDABC under MYDIR.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

        // Open the OracleBFile
        bFile.OpenFile();

        // Search for the 2nd occurrence of a byte pattern {66,67}
        // starting from byte offset 1 in the OracleBFile
        long posFound = bFile.Search(pattern, 1, 2);

        // Prints "posFound = 6"
        Console.WriteLine("posFound = " + posFound);

        // Close the OracleBFile
        bFile.CloseFile();
        bFile.Close();
        bFile.Dispose();

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

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

// Database Setup
/*/ 
-- User must have CREATE ANY DIRECTORY privilege
CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';
/ */

// C#

using System;
using System.IO;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class SeekSample
{
  static void Main()
  {
    // Create MYDIR directory object as indicated above and create a file
    // MyFile.txt with the text ABCDABC under MYDIR.
    // Note that the byte representation of the ABCDABC is 65666768656667

string constr = "User Id=scott;Password=tiger;Data Source=oracle";
OracleConnection con = new OracleConnection(constr);
con.Open();

OracleBFile bFile = new OracleBFile(con, "MYDIR", "MyFile.txt");

// Open the OracleBFile
bFile.OpenFile();

// Set the Position to 2 with respect to SeekOrigin.Begin
long newPosition = bFile.Seek(2, SeekOrigin.Begin);

// Prints "newPosition = 2"
Console.WriteLine("newPosition = " + newPosition);

// Prints "bFile.Position = 2"
Console.WriteLine("bFile.Position = " + bFile.Position);

// Read 2 bytes into readBuffer, starting at buffer offset 1
byte[] readBuffer = new byte[4];
int bytesRead = bFile.Read(readBuffer, 1, 2);

// Prints "bytesRead = 2"
Console.WriteLine("bytesRead = " + bytesRead);

// Prints "readBuffer = 067680"
Console.Write("readBuffer = ");
for(int index = 0; index < readBuffer.Length; index++)
{
    Console.Write("\" + readBuffer[index]);
}
Console.WriteLine();

// Close the OracleBFile
bFile.CloseFile();
bFile.Close();
bFile.Dispose();
con.Close();
con.Dispose();

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleBlobSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();
        OracleBlob blob = new OracleBlob(con);

        // Write 4 bytes from writeBuffer, starting at buffer offset 0
        byte[] writeBuffer = new byte[4] { 1, 2, 3, 4 };
        blob.Write(writeBuffer, 0, 4);

        // Append first 2 bytes from writeBuffer {1, 2} to the oracleBlob
        blob.Append(writeBuffer, 0, 2);

        // Prints "blob.Length = 6"
        Console.WriteLine("blob.Length = " + blob.Length);

        // Reset the Position for the Read
        blob.Position = 0;

        // Read 6 bytes into readBuffer, starting at buffer offset 0
        byte[] readBuffer = new byte[6];
        int bytesRead = blob.Read(readBuffer, 0, 6);

        // Prints "bytesRead = 6"
    }
Console.WriteLine("bytesRead    = " + bytesRead);

// Prints "readBuffer   = 123412"
Console.Write("readBuffer   = ");
for(int index = 0; index < readBuffer.Length; index++)
{
    Console.Write(readBuffer[index]);
}
Console.WriteLine();

// Search for the 2nd occurrence of a byte pattern '12'
// starting from byte offset 0 in the OracleBlob
byte[] pattern = new byte[2] {1, 2};
long posFound = blob.Search(pattern, 0, 2);

// Prints "posFound     = 5"
Console.WriteLine("posFound     = " + posFound);

// Erase 4 bytes of data starting at byte offset 1
// Sets bytes to zero
blob.Erase(1, 4);
byte[] erasedBuffer = blob.Value;

//Prints "erasedBuffer = 100002"
Console.Write("erasedBuffer = ");
for(int index = 0; index < erasedBuffer.Length; index++)
{
    Console.Write(erasedBuffer[index]);
}
Console.WriteLine();

blob.Close();
blob.Dispose();

con.Close();
con.Dispose();
} }
OracleBlob Members

OracleBlob members are listed in the following tables:

OracleBlob Constructors
OracleBlob constructors are listed in Table 8–10.

**Table 8–10  OracleBlob Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleBlob</td>
<td>Creates instance of the OracleBlob class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleBlob Static Fields
OracleBlob static fields are listed in Table 8–11.

**Table 8–11  OracleBlob Static Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSize</td>
<td>Holds maximum number of bytes a BLOB can hold, which is 4,294,967,295 (2^32 - 1) bytes</td>
</tr>
</tbody>
</table>

OracleBlob Static Methods
OracleBlob static methods are listed in Table 8–12.

**Table 8–12  OracleBlob Static Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleBlob Instance Properties
OracleBlob instance properties are listed in Table 8–13.

**Table 8–13  OracleBlob Instance Properties**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether forward and backward seek operations be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether the LOB object supports writing</td>
</tr>
<tr>
<td>Connection</td>
<td>Indicates the OracleConnection that is used to retrieve and write BLOB data</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Indicates whether the BLOB is empty or not</td>
</tr>
<tr>
<td>IsInChunkWriteMode</td>
<td>Indicates whether the BLOB has been opened to defer index updates</td>
</tr>
<tr>
<td>IsTemporary</td>
<td>Indicates whether or not the current instance is bound to a temporary BLOB</td>
</tr>
<tr>
<td>Length</td>
<td>Indicates the size of the BLOB data</td>
</tr>
<tr>
<td>OptimumChunkSize</td>
<td>Indicates the minimum number of bytes to retrieve or send from the server during a read or write operation</td>
</tr>
</tbody>
</table>
OracleBlob Members

OracleBlob Instance Methods
OracleBlob instance methods are listed in Table 8–14.

Table 8–14 OracleBlob Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>Appends the supplied data to the current OracleBlob instance (Overloaded)</td>
</tr>
<tr>
<td>BeginChunkWrite</td>
<td>Opens the BLOB</td>
</tr>
<tr>
<td>BeginRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleBlob object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current stream and releases any resources associated with it</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares data referenced by the current instance and that of the supplied object</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies from the current OracleBlob instance to an OracleBlob object (Overloaded)</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated by this object</td>
</tr>
<tr>
<td>EndChunkWrite</td>
<td>Closes the BLOB referenced by the current OracleBlob instance</td>
</tr>
<tr>
<td>EndRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>Erase</td>
<td>Erases data (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td>Not supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializedLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares the LOB data referenced by the two OracleBlobs</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a specified amount of bytes from the ODP.NET LOB Type instance and populates the buffer</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from Stream</td>
</tr>
</tbody>
</table>
### Table 8–14 (Cont.) OracleBlob Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Search</strong></td>
<td>Searches for a binary pattern in the current instance of an OracleBlob</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>Sets the position in the current LOB stream</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Trims or truncates the BLOB value to the specified length</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Inherited from Object</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes the supplied buffer into the OracleBlob</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Inherited from Stream</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBlob Members
OracleBlob Constructors

OracleBlob constructors are listed in Table 8–10.

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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- OracleBlob Class
- OracleBlob Members
OracleBlob Static Fields

OracleBlob static fields are listed in Table 8–15.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSize</td>
<td>Holds the maximum number of bytes a BLOB can hold, which is 4,294,967,295 (2^32 - 1) bytes</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// 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" on page 1-4
- OracleBlob Class
- OracleBlob Members
OracleBlob Static Methods

OracleBlob static methods are listed in Table 8–16.

Table 8–16  OracleBlob Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBlob Class
- OracleBlob Members
OracleBlob Instance Properties

OracleBlob instance properties are listed in Table 8–17.

Table 8–17  OracleBlob Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether forward and backward seek operations be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether the LOB object supports writing</td>
</tr>
<tr>
<td>Connection</td>
<td>Indicates the OracleConnection that is used to retrieve and write BLOB data</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Indicates whether the BLOB is empty or not</td>
</tr>
<tr>
<td>IsInChunkWriteMode</td>
<td>Indicates whether the BLOB has been opened to defer index updates</td>
</tr>
<tr>
<td>IsTemporary</td>
<td>Indicates whether or not the current instance is bound to a temporary BLOB</td>
</tr>
<tr>
<td>Length</td>
<td>Indicates the size of the BLOB data</td>
</tr>
<tr>
<td>OptimumChunkSize</td>
<td>Indicates the minimum number of bytes to retrieve or send from</td>
</tr>
<tr>
<td></td>
<td>the server during a read or write operation</td>
</tr>
<tr>
<td>Position</td>
<td>Indicates the current read or write position in the LOB stream</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the data, starting from the first byte in BLOB, as a byte array</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBlob Class
- OracleBlob Members

CanRead

Overrides Stream

This instance property indicates whether the LOB stream can be read.

Declaration

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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.
OracleBlob Instance Properties

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBlob Class
- OracleBlob Members

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" on page 1-4
- 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;}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBlob Class
- OracleBlob Members
**Property Value**

bool

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleBlob Class
- OracleBlob Members
OracleBlob Instance Properties

**Position**

Overrides Stream

This instance property indicates the current read or write position in the LOB stream.

**Declaration**

```csharp
// 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" on page 1-4
- OracleBlob Class
- OracleBlob Members

**Value**

This instance property returns the data, starting from the first byte in the BLOB, as a byte array.

**Declaration**

```csharp
// 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" on page 1-4
- OracleBlob Class
- OracleBlob Members
## OracleBlob Instance Methods

OracleBlob instance methods are listed in Table 8–18.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>Appends the supplied data to the current OracleBlob instance (Overloaded)</td>
</tr>
<tr>
<td>BeginChunkWrite</td>
<td>Opens the BLOB</td>
</tr>
<tr>
<td>BeginRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleBlob object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current stream and releases any resources associated with it</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares data referenced by the current instance and that of the supplied object</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies from the current OracleBlob instance to an OracleBlob object (Overloaded)</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated by this object</td>
</tr>
<tr>
<td>EndChunkWrite</td>
<td>Closes the BLOB referenced by the current OracleBlob instance</td>
</tr>
<tr>
<td>EndRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>Erase</td>
<td>Erases data (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td>Not supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializedLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares the LOB data referenced by the two OracleBlobs</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a specified amount of bytes from the ODP.NET LOB Type instance and populates the buffer</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Search</td>
<td>Searches for a binary pattern in the current instance of an OracleBlob</td>
</tr>
<tr>
<td>Seek</td>
<td>Sets the position in the current LOB stream</td>
</tr>
<tr>
<td>SetLength</td>
<td>Trims or truncates the BLOB value to the specified length</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Write</td>
<td>Writes the supplied buffer into the OracleBlob</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Inherited from Stream</td>
</tr>
</tbody>
</table>
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" on page 1-4
- 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.
Append(byte[], int, int)

This instance method appends data from the supplied byte array buffer to the end of the current OracleBlob instance.

Declaration

```csharp
// 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

```csharp
// C#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleBlobSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBlob blob = new OracleBlob(con);

        // Write 4 bytes from writeBuffer, starting at buffer offset 0
        byte[] writeBuffer = new byte[4] {1, 2, 3, 4};
        blob.Write(writeBuffer, 0, 4);

        // Append first 2 bytes from writeBuffer {1, 2} to the oracleBlob
        blob.Append(writeBuffer, 0, 2);

        // Prints "blob.Length = 6"
    }
}
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBlob Class
- OracleBlob Members
Console.WriteLine("blob.length = " + blob.Length);

// Reset the Position for the Read
blob.Position = 0;

// Read 6 bytes into readBuffer, starting at buffer offset 0
byte[] readBuffer = new byte[6];
int bytesRead = blob.Read(readBuffer, 0, 6);

// Prints "bytesRead = 6"
Console.WriteLine("bytesRead = " + bytesRead);

// Prints "readBuffer = 123412"
Console.Write("readBuffer = ");
for(int index = 0; index < readBuffer.Length; index++)
{
    Console.Write(readBuffer[index]);
}
Console.WriteLine();

// Search for the 2nd occurrence of a byte pattern '12'
// starting from byte offset 0 in the OracleBlob
byte[] pattern = new byte[2] {1, 2};
long posFound = blob.Search(pattern, 0, 2);

// Prints "posFound = 5"
Console.WriteLine("posFound = " + posFound);

// Erase 4 bytes of data starting at byte offset 1
// Sets bytes to zero
blob.Erase(1, 4);
byte[] erasedBuffer = blob.Value;

// Prints "erasedBuffer = 100002"
Console.Write("erasedBuffer = ");
for(int index = 0; index < erasedBuffer.Length; index++)
{
    Console.Write(erasedBuffer[index]);
}
Console.WriteLine();

blob.Close();
blob.Dispose();
con.Close();
con.Dispose();
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" on page 1-4
- 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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CloneSample
```csharp
static void Main()
{
    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleBlob blob1 = new OracleBlob(con);

    // Prints "blob1.Position = 0"
    Console.WriteLine("blob1.Position = " + blob1.Position);

    // Set the Position before calling Clone()
    blob1.Position = 1;

    // Clone the OracleBlob
    OracleBlob blob2 = (OracleBlob)blob1.Clone();

    // Prints "blob2.Position = 1"
    Console.WriteLine("blob2.Position = " + blob2.Position);

    blob1.Close();
    blob1.Dispose();

    blob2.Close();
    blob2.Dispose();

    con.Close();
    con.Dispose();
}
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleBlob Class
- OracleBlob Members
**Compare**

This instance method compares data referenced by the current instance and that of the supplied object.

**Declaration**

```csharp
// 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.

- `InvalidOperation` - 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" on page 1-4

- `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" on page 1-4
- OracleBlob Class
- OracleBlob Members

**CopyTo(OracleBlob)**

This instance method copies data from the current instance to the provided OracleBlob object.

**Declaration**

```csharp
// 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.
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" on page 1-4
- 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**

```csharp
// 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`, `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**

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

class CopyToSample
{
    static void Main()
```
{  
    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleBlob blob1 = new OracleBlob(con);
    OracleBlob blob2 = new OracleBlob(con);

    // Write 4 bytes, starting at buffer offset 0
    byte[] buffer = new byte[4] {1, 2, 3, 4};
    blob1.Write(buffer, 0, 4);

    // Copy 2 bytes from byte 0 of blob1 to byte 1 of blob2
    blob1.CopyTo(0, blob2, 1, 2);

    byte[] copyBuffer = blob2.Value;

    //Prints "Value = 012"
    Console.Write("Value = ");
    for(int index = 0; index < copyBuffer.Length; index++)
    {
        Console.Write(copyBuffer[index]);
    }
    Console.WriteLine();

    blob1.Close();
    blob1.Dispose();

    blob2.Close();
    blob2.Dispose();

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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.

**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" on page 1-4
- OracleBlob Class
- OracleBlob Members
**IsEqual**

This instance method compares the LOB data referenced by the two *OracleBlob* objects.

**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" on page 1-4
- *OracleBlob* Class
- *OracleBlob* Members

**Read**

Overides 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#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class ReadSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBlob blob = new OracleBlob(con);

        // Write 3 bytes, starting at buffer offset 1
        byte[] writeBuffer = new byte[4] {1, 2, 3, 4};
        blob.Write(writeBuffer, 1, 3);

        // Reset the Position for Read
        blob.Position = 1;

        // Read 2 bytes into buffer starting at buffer offset 1
        byte[] readBuffer = new byte[4];
        int bytesRead = blob.Read(readBuffer, 1, 2);

        // Prints 'bytesRead = 2'
        Console.WriteLine("bytesRead = " + bytesRead);

        // Prints 'readBuffer = 0340'
        Console.Write("readBuffer = ");
        for(int index = 0; index < readBuffer.Length; index++)
        {
            Console.Write(readBuffer[index]);
        }
    }
}
Search

This instance method searches for a binary pattern in the current instance of an OracleBlob.

Declaration

```csharp
// 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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class SearchSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleBlob blob = new OracleBlob(con);

        // Write 7 bytes, starting at buffer offset 0
        byte[] buffer = new byte[7] {1, 2, 3, 4, 1, 2, 3};
        blob.Write(buffer, 0, 7);

        // Search for the 2nd occurrence of a byte pattern '23'
        // starting at offset 1 in the OracleBlob
        byte[] pattern = new byte[2] {2, 3};
        long posFound = blob.Search(pattern, 1, 2);

        // Prints "posFound = 6"
        Console.WriteLine("posFound = "+ posFound);

        blob.Close();
        blob.Dispose();

        con.Close();
        con.Dispose();
    }
}
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBlob Class
- OracleBlob 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);
```
OracleBlob Instance Methods

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" on page 1-4
- 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.
OracleBlob Class

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" on page 1-4
- 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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class WriteSample
```csharp
{ static void Main()
{
    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleBlob blob = new OracleBlob(con);

    // Set the Position for the Write
    blob.Position = 0;

    // Begin ChunkWrite to improve performance
    // Index updates occur only once after EndChunkWrite
    blob.BeginChunkWrite();

    // Write to the OracleBlob in 5 chunks of 2 bytes each
    byte[] b = new byte[2] {1, 2};
    for(int index = 0; index < 5; index++)
    {
        blob.Write(b, 0, b.Length);
    }
    blob.EndChunkWrite();

    byte[] chunkBuffer = blob.Value;

    // Prints "chunkBuffer = 1212121212"
    Console.Write("chunkBuffer = ");
    for(int index = 0; index < chunkBuffer.Length; index++)
    {
        Console.Write(chunkBuffer[index]);
    }
    Console.WriteLine();

    blob.Close();
    blob.Dispose();

    con.Close();
    con.Dispose();
}
}
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleClobSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob = new OracleClob(con);

        // Write 4 chars from writeBuffer, starting at buffer offset 0
        char[] writeBuffer = new char[4] {'a', 'b', 'c', 'd'};
        clob.Write(writeBuffer, 0, 4);

        // Append first 2 chars from writeBuffer {'a', 'b'} to the oracleClob
        clob.Append(writeBuffer, 0, 2);

        // Prints 'clob.Length = 12'
        Console.WriteLine("clob.Length = " + clob.Length);

        // Reset the Position for the Read
        clob.Position = 0;
    }
}
/ Read 6 chars into readBuffer, starting at buffer offset 0
char[] readBuffer = new char[6];
int charsRead = clob.Read(readBuffer, 0, 6);

// Prints "charsRead = 6"
Console.WriteLine("charsRead = ' + charsRead);

// Prints "readBuffer = abcdab"
Console.Write("readBuffer = ");
for(int index = 0; index < readBuffer.Length; index++)
{
    Console.Write(readBuffer[index]);
}
Console.WriteLine();

// Search for the 2nd occurrence of a char pattern 'ab'
// starting from char offset 0 in the OracleClob
char[] pattern = new char[2] {'a', 'b'};
long posFound = clob.Search(pattern, 0, 2);

// Prints "posFound = 5"
Console.WriteLine("posFound = ' + posFound);

// Erase 4 chars of data starting at char offset 1
// Sets chars to ''
clob.Erase(1, 4);

// Prints "clob.Value = a    b"
Console.WriteLine("clob.Value = ' + clob.Value);

clob.Close();
clob.Dispose();

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

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll

See Also:
■ "Oracle.DataAccess.Types Namespace" on page 1-4
■ 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 8–19.

**Table 8–19  OracleClob Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleClob Constructors</td>
<td>Creates an instance of the OracleClob class bound to a temporary CLOB (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleClob Static Fields**
OracleClob static fields are listed in Table 8–20.

**Table 8–20  OracleClob Static Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSize</td>
<td>Holds the maximum number of bytes a CLOB can hold, which is 4,294,967,295 (2^32 - 1) bytes</td>
</tr>
</tbody>
</table>

**OracleClob Static Methods**
OracleClob static methods are listed in Table 8–21.

**Table 8–21  OracleClob Static Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleClob Instance Properties**
OracleClob instance properties are listed in Table 8–22.

**Table 8–22  OracleClob Instance Properties**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether forward and backward seek operations can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether the LOB stream can be written</td>
</tr>
<tr>
<td>Connection</td>
<td>Indicates the OracleConnection that is used to retrieve and write CLOB data</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Indicates whether the CLOB is empty or not</td>
</tr>
<tr>
<td>IsInChunkWriteMode</td>
<td>Indicates whether or not the CLOB has been opened</td>
</tr>
<tr>
<td>IsNCLOB</td>
<td>Indicates whether the OracleClob object represents an NCLOB.</td>
</tr>
<tr>
<td>IsTemporary</td>
<td>Indicates whether or not the current instance is bound to a temporary CLOB</td>
</tr>
<tr>
<td>Length</td>
<td>Indicates the size of the CLOB data in bytes</td>
</tr>
</tbody>
</table>
Table 8–22 (Cont.) OracleClob Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptimumChunkSize</td>
<td>Indicates the minimum number of bytes to retrieve or send from the server during a read or write operation</td>
</tr>
<tr>
<td>Position</td>
<td>Indicates the current read or write position in the LOB stream in bytes</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the data, starting from the first character in the CLOB or NCLOB, as a string</td>
</tr>
</tbody>
</table>

OracleClob Instance Methods

The OracleClob instance methods are listed in Table 8–23.

Table 8–23 OracleClob Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>Appends data to the current OracleClob instance (Overloaded)</td>
</tr>
<tr>
<td>BeginChunkWrite</td>
<td>Opens the CLOB</td>
</tr>
<tr>
<td>BeginRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleClob object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current stream and releases resources associated with it</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares data referenced by the current instance to that of the supplied object</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies the data to an OracleClob (Overloaded)</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated by this object</td>
</tr>
<tr>
<td>EndChunkWrite</td>
<td>Closes the CLOB referenced by the current OracleClob instance</td>
</tr>
<tr>
<td>EndRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>Erase</td>
<td>Erases the specified amount of data (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td>Not supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the current instance</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares the LOB data referenced by two OracleClobs</td>
</tr>
<tr>
<td>Read</td>
<td>Reads from the current instance (Overloaded)</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from Stream</td>
</tr>
</tbody>
</table>
### Table 8–23  (Cont.) OracleClob Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>Searches for a character pattern in the current instance of OracleClob (Overloaded)</td>
</tr>
<tr>
<td>Seek</td>
<td>Sets the position in the current LOB stream</td>
</tr>
<tr>
<td>SetLength</td>
<td>Trims or truncates the CLOB value</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Write</td>
<td>Writes the provided buffer into the OracleClob (Overloaded)</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Inherited from Stream</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- 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" on page 1-4
- OracleClob Class
- OracleClob Members
OracleClob Static Fields

OracleClob static fields are listed in Table 8–24.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSize</td>
<td>Holds the maximum number of bytes a CLOB can hold, which is 4,294,967,295 (2^32 - 1) bytes</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// 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" on page 1-4
- OracleClob Class
- OracleClob Members
OracleClob Static Methods

OracleClob static methods are listed in Table 8–25.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from <code>Object</code> (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleClob Class
- OracleClob Members
OracleClob Instance Properties

OracleClob instance properties are listed in Table 8–26.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether forward and backward seek operations can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether the LOB stream can be written</td>
</tr>
<tr>
<td>Connection</td>
<td>Indicates the OracleConnection that is used to retrieve and write CLOB data</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Indicates whether the CLOB is empty or not</td>
</tr>
<tr>
<td>IsInChunkWriteMode</td>
<td>Indicates whether or not the CLOB has been opened</td>
</tr>
<tr>
<td>IsNCLOB</td>
<td>Indicates whether the OracleClob object represents an NCLOB.</td>
</tr>
<tr>
<td>IsTemporary</td>
<td>Indicates whether or not the current instance is bound to a temporary CLOB</td>
</tr>
<tr>
<td>Length</td>
<td>Indicates the size of the CLOB data in bytes</td>
</tr>
<tr>
<td>OptimumChunkSize</td>
<td>Indicates the minimum number of bytes to retrieve or send from the server during a read or write operation</td>
</tr>
<tr>
<td>Position</td>
<td>Indicates the current read or write position in the LOB stream in bytes</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the data, starting from the first character in the CLOB or NCLOB, as a string</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleClob Class
- OracleClob Members

CanRead

Overrides Stream

This instance property indicates whether the LOB stream can be read.

Declaration

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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.
OracleClob Instance Properties

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleClob Class
- OracleClob Members

IsNCLOB
This instance property indicates whether the OracleClob object represents an NCLOB.

Declaration
// C#
public bool IsNCLOB {get;}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleClob Class
- OracleClob Members
### Property Value

A `bool`.

#### See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleClob Class
- OracleClob Members

### IsTemporary

This instance property indicates whether or not the current instance is bound to a temporary CLOB.

#### Declaration

```csharp
// C#
public bool IsTemporary {get;}
```

#### Property Value

A `bool`.

#### See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleClob Class
- OracleClob Members

### Length

Overides Stream

This instance property indicates the size of the CLOB data in bytes.

#### Declaration

```csharp
// 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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- OracleClob Class
- OracleClob Members

**Position**

Overrides `Stream`

This instance property indicates the current read or write position in the LOB stream in bytes.

**Declaration**

```csharp
// 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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- OracleClob Class
- OracleClob Members
OracleClob Instance Methods

The OracleClob instance methods are listed in Table 8–27.

Table 8–27 OracleClob Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>Appends data to the current OracleClob instance (Overloaded)</td>
</tr>
<tr>
<td>BeginChunkWrite</td>
<td>Opens the CLOB</td>
</tr>
<tr>
<td>BeginRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleClob object</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current stream and releases resources associated with it</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares data referenced by the current instance to that of the supplied object</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies the data to an OracleClob (Overloaded)</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated by this object</td>
</tr>
<tr>
<td>EndChunkWrite</td>
<td>Closes the CLOB referenced by the current OracleClob instance</td>
</tr>
<tr>
<td>EndRead</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>Erase</td>
<td>Erases the specified amount of data (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td>Not supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the current instance</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from MarshalByRefObject</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares the LOB data referenced by two OracleClobs</td>
</tr>
<tr>
<td>Read</td>
<td>Reads from the current instance (Overloaded)</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from Stream</td>
</tr>
<tr>
<td>Search</td>
<td>Searches for a character pattern in the current instance of OracleClob (Overloaded)</td>
</tr>
<tr>
<td>Seek</td>
<td>Sets the position in the current LOB stream</td>
</tr>
<tr>
<td>SetLength</td>
<td>Trims or truncates the CLOB value</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>Write</td>
<td>Writes the provided buffer into the OracleClob (Overloaded)</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Inherited from Stream</td>
</tr>
</tbody>
</table>
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" on page 1-4
■ 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" on page 1-4
- `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" on page 1-4
- `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#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class AppendSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob = new OracleClob(con);

        // Append 2 chars {'d', 'e'} to the OracleClob
        char[] buffer = new char[3] {'d', 'e', 'f'};
        clob.Append(buffer, 0, 2);

        // Prints 'clob.Value = de'
        Console.WriteLine("clob.Value = " + clob.Value);

        clob.Close();
        clob.Dispose();

        con.Close();
        con.Dispose();
    }
}
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CloneSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob1 = new OracleClob(con);

        // Prints "clob1.Position = 0"
        Console.WriteLine("clob1.Position = " + clob1.Position);

        // Set the Position before calling Clone()
        clob1.Position = 1;

        // Clone the OracleClob
        OracleClob clob2 = (OracleClob)clob1.Clone();

        // Prints "clob2.Position = 1"
        Console.WriteLine("clob2.Position = " + clob2.Position);

        clob1.Close();
        clob1.Dispose();

        clob2.Close();
        clob2.Dispose();
    }
}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleClob Class
- OracleClob Members

CopyTo(OracleClob)

This instance method copies data from the current instance to the provided OracleClob object.

Declaration

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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**

```csharp
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.
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class CopyToSample
{
  static void Main()
  {
    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    OracleClob clob1 = new OracleClob(con);
    OracleClob clob2 = new OracleClob(con);

    // Write 4 chars, starting at buffer offset 0
    char[] buffer = new char[4] {'a', 'b', 'c', 'd'};
    clob1.Write(buffer, 0, 4);

    // Copy 2 chars from char 0 of clob1 to char 1 of clob2
    clob1.CopyTo(0, clob2, 1, 2);

    // Prints "clob2.Value = ab"
    Console.WriteLine("clob2.Value = " + clob2.Value);

    clob1.Close();
    clob1.Dispose();

    clob2.Close();
    clob2.Dispose();
    con.Close();
    con.Dispose();
  }
}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleClob Class
- OracleClob Members

GetHashCode

Overrides Object
This method returns a hash code for the current instance.

**Declaration**

```csharp
// C#  
public override int GetHashCode();
```

**Return Value**

An `int` representing a hash code.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleClob Class
- OracleClob Members

**IsEqual**

This instance method compares the LOB data referenced by two `OracleClob` objects.

**Declaration**

```csharp
// 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" on page 1-4
- 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" on page 1-4
  - OracleClob Class
  - OracleClob Members

**Read(byte [], int, int)**

Overides Stream

This instance method reads a specified amount of bytes from the current instance and populates the byte array buffer.

**Declaration**

```csharp
// 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.
**Read(char[], int, int)**

This instance method reads a specified amount of characters from the current instance and populates the character array buffer.

**Declaration**

```csharp
// 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**

```csharp
// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class ReadSample
```
```csharp
{ static void Main()
{ string constr = "User Id=scott;Password=tiger;Data Source=oracle";
OracleConnection con = new OracleConnection(constr);
con.Open();

OracleClob clob = new OracleClob(con);

// Write 3 chars, starting at buffer offset 1
char[] writeBuffer = new char[4] {'a', 'b', 'c', 'd'};
clob.Write(writeBuffer, 1, 3);

// Reset the Position (in bytes) for Read
clob.Position = 2;

// Read 2 chars into buffer starting at buffer offset 1
char[] readBuffer = new char[4];
int charsRead = clob.Read(readBuffer, 1, 2);

// Prints "charsRead = 2"
Console.WriteLine("charsRead = " + charsRead);

// Prints "readBuffer = cd "
Console.Write("readBuffer = ");
for(int index = 0; index < readBuffer.Length; index++)
{ Console.Write(readBuffer[index]);
} Console.WriteLine();

clob.Close();
clob.Dispose();

con.Close();
con.Dispose();
}
}
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
OracleClob Class

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.
Search(char[], Int64, Int64)

This instance method searches for a character pattern in the current instance of OracleClob.

Declaration

```csharp
// 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

```csharp
// C#
```
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class SearchSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob = new OracleClob(con);

        // Write 7 chars, starting at buffer offset 0
        char[] buffer = new char[7] {'a', 'b', 'c', 'd', 'a', 'b', 'c'};
        clob.Write(buffer, 0, 7);

        // Search for the 2nd occurrence of a char pattern 'bc'
        // starting at offset 1 in the OracleBlob
        char[] pattern = new char[2] {'b', 'c'};
        long posFound = clob.Search(pattern, 1, 2);

        // Prints *posFound = 6*
        Console.WriteLine("posFound = " + posFound);

        clob.Close();
        clob.Dispose();

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleClob Class
- OracleClob 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 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" on page 1-4
- 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.
OracleClob Class

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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.
OracleClob Class

- 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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class WriteSample
{
    static void Main()
    {
        string constr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(constr);
        con.Open();

        OracleClob clob = new OracleClob(con);

        // Set the Position for the Write;
        clob.Position = 0;

        // Begin ChunkWrite to improve performance
        // Index updates occur only once after EndChunkWrite
        clob.BeginChunkWrite();

        // Write to the OracleClob in 5 chunks of 2 chars each
        char[] c = new char[2] {'a', 'b'};
        for (int index = 0; index < 5; index++)
        {
            clob.Write(c, 0, c.Length);
        }
        clob.EndChunkWrite();

        // Prints "clob.Value = ababababab"
        Console.WriteLine("clob.Value = " + clob.Value);

        clob.Close();
        clob.Dispose();

        con.Close();
        con.Dispose();
    }
}
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

// Database Setup
/*
CREATE OR REPLACE FUNCTION MyFunc(refcur_out OUT SYS_REFCURSOR)
  RETURN SYS_REFCURSOR IS refcur_ret SYS_REFCURSOR;
BEGIN
  OPEN refcur_ret FOR SELECT * FROM EMP;
  OPEN refcur_out FOR SELECT * FROM DEPT;
  RETURN refcur_ret;
END MyFunc;
*/

using System;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;
class OracleRefCursorSample
{
  static void Main()
  {
    // Example demonstrates how to use REF CURSORs returned from
    // PL/SQL Stored Procedures or Functions
    // Create the PL/SQL Function MyFunc as defined above

    string constr = "User Id=scott;Password=tiger;Data Source=oracle";
    OracleConnection con = new OracleConnection(constr);
    con.Open();

    // Create an OracleCommand
    OracleCommand cmd = new OracleCommand("MyFunc", con);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add(OracleDbType.RefCursor, "refcur_out");
    cmd.ExecuteNonQuery();

    // Bind the parameters
// p1 is the RETURN REF CURSOR bound to SELECT * FROM EMP;
OracleParameter p1 =
    cmd.Parameters.Add("refcur_ret", OracleDbType.RefCursor);
p1.Direction = ParameterDirection.ReturnValue;

// p2 is the OUT REF CURSOR bound to SELECT * FROM DEPT
OracleParameter p2 =
    cmd.Parameters.Add("refcur_out", OracleDbType.RefCursor);
p2.Direction = ParameterDirection.Output;

// Execute the command
cmd.ExecuteNonQuery();

// Construct an OracleDataReader from the REF CURSOR
OracleDataReader reader1 = ((OracleRefCursor)p1.Value).GetDataReader();

// Prints "reader1.GetName(0) = EMPNO"
Console.WriteLine("reader1.GetName(0) = " + reader1.GetName(0));

// Construct an OracleDataReader from the REF CURSOR
OracleDataReader reader2 = ((OracleRefCursor)p2.Value).GetDataReader();

// Prints "reader2.GetName(0) = DEPTNO"
Console.WriteLine("reader2.GetName(0) = " + reader2.GetName(0));

reader1.Close();
reader1.Dispose();

reader2.Close();
reader2.Dispose();

p1.Dispose();
p2.Dispose();

cmd.Dispose();

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

Requirements

Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll

See Also:
  ■ "Oracle.DataAccess.Types Namespace" on page 1-4
  ■ 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 8–28.

*Table 8–28  OracleRefCursor Static Methods*

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleRefCursor Properties**
OracleRefCursor properties are listed in Table 8–29.

*Table 8–29  OracleRefCursor Properties*

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>A reference to the OracleConnection used to fetch the REF CURSOR data</td>
</tr>
</tbody>
</table>

**OracleRefCursor Instance Methods**
OracleRefCursor instance methods are listed in Table 8–30.

*Table 8–30  OracleRefCursor Instance Methods*

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the resources allocated by the OracleRefCursor object</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetDataReader</td>
<td>Returns an OracleDataReader object for the REF CURSOR</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleRefCursor Class
OracleRefCursor Static Methods

OracleRefCursor static methods are listed in Table 8–31.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleRefCursor Class
- OracleRefCursor Members
OracleRefCursor Properties

OracleRefCursor properties are listed in Table 8–32.

Table 8–32  OracleRefCursor Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>A reference to the OracleConnection used to fetch the REF CURSOR data</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleRefCursor Class
- OracleRefCursor Members

Connection

This property refers to the OracleConnection used to fetch the REF CURSOR data.

Declaration

```csharp
// 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" on page 1-4
- OracleRefCursor Class
- OracleRefCursor Members
OracleRefCursor Instance Methods

OracleRefCursor instance methods are listed in Table 8–33.

### Table 8–33 OracleRefCursor Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the resources allocated by the OracleRefCursor object</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetDataReader</td>
<td>Returns an OracleDataReader object for the REF CURSOR</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleRefCursor Class
- OracleRefCursor Members

### Dispose

This instance method disposes of the resources allocated by the OracleRefCursor object.

**Declaration**

```csharp
// 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" on page 1-4
- OracleRefCursor Class
- OracleRefCursor Members

### GetDataReader

This instance method returns an OracleDataReader object for the REF CURSOR.

**Declaration**

```csharp
// 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" on page 1-4
- OracleRefCursor Class
- OracleRefCursor Members
This chapter describes the Oracle Data Provider for .NET Types structures.

This chapter contains these topics:

- OracleBinary Structure
- OracleDate Structure
- OracleDecimal Structure
- OracleIntervalDS Structure
- OracleIntervalYM Structure
- OracleString Structure
- OracleTimeStamp Structure
- OracleTimeStampLTZ Structure
- OracleTimeStampTZ Structure
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#
   using System;
   using Oracle.DataAccess.Types;
   
   class OracleBinarySample
   {
      static void Main(string[] args)
      {
         // Initialize the OracleBinary structures
         OracleBinary binary1 = new OracleBinary(new byte[] {1,2,3,4,5});
         OracleBinary binary2 = new OracleBinary(new byte[] {1,2,3});
         OracleBinary binary3 = new OracleBinary(new byte[] {4,5});
         OracleBinary binary4 = binary2 + binary3;

         // Compare binary1 and binary4; they're equal
         if (binary1 == binary4)
            Console.WriteLine("The two OracleBinary structs are equal");
         else
            Console.WriteLine("The two OracleBinary structs are different");
      }
   }

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 9–1.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleBinary Constructor</td>
<td>Instantiates a new instance of OracleBinary structure</td>
</tr>
</tbody>
</table>

**OracleBinary Static Fields**
The OracleBinary static fields are listed in Table 9–2.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleBinary structure</td>
</tr>
</tbody>
</table>

**OracleBinary Static Methods**
The OracleBinary static methods are listed in Table 9–3.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concat</td>
<td>Returns the concatenation of two OracleBinary structures</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines if two OracleBinary values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleBinary values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleBinary values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleBinary values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleBinary values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleBinary values are not equal</td>
</tr>
</tbody>
</table>

**OracleBinary Static Operators**
The OracleBinary static operators are listed in Table 9–4.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Concatenates two OracleBinary values</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleBinary values are equal</td>
</tr>
</tbody>
</table>
OracleBinary Static Type Conversion Operators
The OracleBinary static type conversion operators are listed in Table 9–5.

Table 9–5  OracleBinary Static Type Conversion Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator byte[ ]</td>
<td>Converts an instance value to a byte array</td>
</tr>
<tr>
<td>implicit operator OracleBinary</td>
<td>Converts an instance value to an OracleBinary structure</td>
</tr>
</tbody>
</table>

OracleBinary Properties
The OracleBinary properties are listed in Table 9–6.

Table 9–6  OracleBinary Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>Item</td>
<td>Obtains the particular byte in an OracleBinary structure using an index</td>
</tr>
<tr>
<td>Length</td>
<td>Returns the length of the binary data</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the binary data that is stored in an OracleBinary structure</td>
</tr>
</tbody>
</table>

OracleBinary Instance Methods
The OracleBinary instance methods are listed in Table 9–7.

Table 9–7  OracleBinary Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current instance to an object and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines if two objects contain the same binary data (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the current instance</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleBinary structure to a string</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- `OracleBinary Structure`
- `OracleBinary Members`
OracleBinary Static Fields

The OracleBinary static fields are listed in Table 9–8.

Table 9–8  OracleBinary Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleBinary structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleBinary Structure
- OracleBinary Members
OracleBinary Static Methods

The OracleBinary static methods are listed in Table 9–9.

Table 9–9 OracleBinary Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concat</td>
<td>Returns the concatenation of two OracleBinary structures</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines if two OracleBinary values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleBinary values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleBinary values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleBinary values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleBinary values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleBinary values are not equal</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBinary Structure
- OracleBinary Members

Concat

This method returns the concatenation of two OracleBinary structures.

Declaration

```csharp
// 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.
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 OracleBinarys that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBinary Structure
- OracleBinary Members

GreaterThanOr

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.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBinary Structure
- OracleBinary Members
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 OracleBinarys that contain a null value are equal.

Example

// C#

using System;
using Oracle.DataAccess.Types;

class GreaterThanSample
{
    static void Main(string[] args)
    {
        OracleBinary binary1 = OracleBinary.Null;
        OracleBinary binary2 = new OracleBinary(new byte[] {1});

        // Compare two OracleBinary structs; binary1 < binary2
        if (OracleBinary.GreaterThan(binary1, binary2))
            Console.WriteLine("binary1 > binary2");
        else
            Console.WriteLine("binary1 < binary2");
    }
}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- `OracleBinary` Structure
- `OracleBinary` Members

---

**LessThan**

This method determines whether the first of two `OracleBinary` values is less than the second.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleBinary` Structure
- `OracleBinary` Members
LessThanOrEqual

This method determines whether the first of two `OracleBinary` values is less than or equal to the second.

Declaration

```csharp
// 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" on page 1-4
- `OracleBinary Structure`
- `OracleBinary Members`

NotEquals

This method determines whether two `OracleBinary` values are not equal.

Declaration

```csharp
// 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 OracleBinarys that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBinary Structure
- OracleBinary Members
OracleBinary Static Operators

The OracleBinary static operators are listed in Table 9–10.

### Table 9–10  OracleBinary Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Concatenates two OracleBinary values</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleBinary values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleBinary values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleBinary values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if two OracleBinary values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleBinary value is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleBinary value is less than or equal to the second</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBinary Structure
- OracleBinary Members

**operator +**

This method concatenates two OracleBinary values.

**Declaration**

```csharp
// 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 OracleBinary structure has a null value.
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" on page 1-4
- 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.
**OracleBinary Structure**

**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` that contain a null value are equal.

**Example**

```csharp
// C#
using System;
using Oracle.DataAccess.Types;

class OperatorSample
{
    static void Main(string[] args)
    {
        OracleBinary binary1 = OracleBinary.Null;
        OracleBinary binary2 = new OracleBinary(new byte[] {1});

        // Compare two OracleBinary structs; binary1 < binary2
        if (binary1 > binary2)
            Console.WriteLine("binary1 > binary2");
        else
            Console.WriteLine("binary1 < binary2");
    }
}
```

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleBinary Structure`
- `OracleBinary Members`

**operator >=**

This method determines if the first of two `OracleBinary` values is greater than or equal to the second.

**Declaration**

```csharp
// C#
public static bool operator >= (OracleBinary value1, OracleBinary value2);
```

**Parameters**

- `value1`
  
  First `OracleBinary`.

- `value2`
  
  Second `OracleBinary`. 
Oracle Binary Static Operators

**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` values that contain a null value are equal.

**See Also:**

- `OracleBinary Structure`
- `OracleBinary Members`

operator !=

This method determines if two `OracleBinary` values are not equal.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleBinary Structure`
- `OracleBinary Members`

operator <

This method determines if the first of two `OracleBinary` values is less than the second.

**Declaration**

```csharp
// 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 OracleBinaries that contain a null value are equal.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBinary Structure
- OracleBinary Members

**operator <=**
This method determines if the first of two OracleBinary values is less than or equal to the second.

**Declaration**
```csharp
// 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 OracleBinaries that contain a null value are equal.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleBinary Structure
- OracleBinary Members
OracleBinary Static Type Conversion Operators

The OracleBinary static type conversion operators are listed in Table 9–11.

Table 9–11  OracleBinary Static Type Conversion Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator byte[ ]</td>
<td>Converts an instance value to a byte array</td>
</tr>
<tr>
<td>implicit operator OracleBinary</td>
<td>Converts an instance value to an OracleBinary structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleBinary Structure
- OracleBinary Members
OracleBinary Properties

The OracleBinary properties are listed in Table 9–12.

Table 9–12  OracleBinary Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>Item</td>
<td>Obtains the particular byte in an OracleBinary structure using an index</td>
</tr>
<tr>
<td>Length</td>
<td>Returns the length of the binary data</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the binary data that is stored in an OracleBinary structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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#

using System;
using Oracle.DataAccess.Types;

class ItemSample
{
    static void Main(string[] args)
    {
        OracleBinary binary = new OracleBinary(new byte[] {1,2,3,4});

        // Prints the value 4
        Console.WriteLine(binary[binary.Length - 1]);
    }
}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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#

using System;
using Oracle.DataAccess.Types;

class LengthSample
{
    static void Main(string[] args)
    {
        OracleBinary binary = new OracleBinary(new byte[] {1,2,3,4});

        // Prints the value 4
        Console.WriteLine(binary.Length);
    }
}
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" on page 1-4
- OracleBinary Structure
- OracleBinary Members
OracleBinary Instance Methods

The OracleBinary instance methods are listed in Table 9–13.

Table 9–13 OracleBinary Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current instance to an object and returns an integer that</td>
</tr>
<tr>
<td></td>
<td>represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines if two objects contain the same binary data (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the current instance</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleBinary structure to a string</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
OracleBinary Structure

Remarks
The following rules apply to the behavior of this method.

■ The comparison must be between OracleBinaries. 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 OracleBinaries that contain a null value are equal.

Example

// C#

using System;
using Oracle.DataAccess.Types;

class CompareToSample
{
    static void Main(string[] args)
    {
        OracleBinary binary1 = new OracleBinary(new byte[] {1,2,3});
        OracleBinary binary2 = new OracleBinary(new byte[] {1,2,3,4});

        // Compare
        if (binary1.CompareTo(binary2) == 0)
            Console.WriteLine("binary1 is the same as binary2");
        else
            Console.WriteLine("binary1 is different from binary2");
    }
}

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-4
■ 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` instances that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleBinary` Structure
- `OracleBinary` Members

GetHashCode
Overrides `Object`
This method returns a hash code for the `OracleBinary` instance.

Declaration

```csharp
// C#
public override int GetHashCode();
```

Return Value
An `int` that represents the hash.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleBinary` Structure
- `OracleBinary` Members

ToString
Overrides `Object`
This method converts an `OracleBinary` instance to a string instance.

Declaration

```csharp
// C#
public override string ToString();
```

Return Value
A `string`.

Remarks
If the current `OracleBinary` instance has a null value, the returned string is "null".

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleDateSample
{
    static void Main(string[] args)
    {
        // Initialize the dates to the lower and upper boundaries
        OracleDate date1 = OracleDate.MinValue;
        OracleDate date2 = OracleDate.MaxValue;
        OracleDate date3 = new OracleDate(DateTime.MinValue);
        OracleDate date4 = new OracleDate(DateTime.MaxValue);

        // Set the thread’s DateFormat for output
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "DD-MON-YYYY BC";
        OracleGlobalization.SetThreadInfo(info);

        // Print the lower and upper boundaries
        Console.WriteLine("OracleDate ranges from\n{n}[0]\nto\n{n}[1]\n\n", date1, date2);
        Console.WriteLine(".NET DateTime ranges from\n{n}[0]\nto\n{n}[1]\n\n", date3, date4);
    }
}

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 9–14.

Table 9–14  OracleDate Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleDate Constructors</td>
<td>Instantiates a new instance of OracleDate structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleDate Static Fields
The OracleDate static fields are listed in Table 9–15.

Table 9–15  OracleDate Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an OracleDate structure, which is December 31, 9999 23:59:59</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an OracleDate structure, which is January 1, -4712 0:0:0</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to the value of an OracleDate structure instance</td>
</tr>
</tbody>
</table>

OracleDate Static Methods
The OracleDate static methods are listed in Table 9–16.

Table 9–16  OracleDate Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleDate values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleDate values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleDate values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleDate values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleDate values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleDate values are not equal</td>
</tr>
<tr>
<td>GetSysDate</td>
<td>Returns an OracleDate structure that represents the current date and time</td>
</tr>
<tr>
<td>Parse</td>
<td>Returns an OracleDate structure and sets its value using a string</td>
</tr>
</tbody>
</table>

OracleDate Static Operators
The OracleDate static operators are listed in Table 9–17.
OracleDate Members

Table 9–17 OracleDate Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator ==</td>
<td>Determines if two OracleDate values are the same</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleDate values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleDate values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if the two OracleDate values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleDate values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleDate values is less than or equal to the second</td>
</tr>
</tbody>
</table>

OracleDate Static Type Conversions
The OracleDate static type conversions are listed in Table 9–18.

Table 9–18 OracleDate Static Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator DateTime</td>
<td>Converts a structure to a DateTime structure</td>
</tr>
<tr>
<td>explicit operator OracleDate</td>
<td>Converts a structure to an OracleDate structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleDate Properties
The OracleDate properties are listed in Table 9–19.

Table 9–19 OracleDate Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Gets an array of bytes that represents an Oracle DATE in Oracle internal format</td>
</tr>
<tr>
<td>Day</td>
<td>Gets the day component of an OracleDate method</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>Hour</td>
<td>Gets the hour component of an OracleDate</td>
</tr>
<tr>
<td>Minute</td>
<td>Gets the minute component of an OracleDate</td>
</tr>
<tr>
<td>Month</td>
<td>Gets the month component of an OracleDate</td>
</tr>
<tr>
<td>Second</td>
<td>Gets the second component of an OracleDate</td>
</tr>
<tr>
<td>Value</td>
<td>Gets the date and time that is stored in the OracleDate structure</td>
</tr>
<tr>
<td>Year</td>
<td>Gets the year component of an OracleDate</td>
</tr>
</tbody>
</table>

OracleDate Methods
The OracleDate methods are listed in Table 9–20.
### OracleDate Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleDate instance to an object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same date and time as the current OracleDate instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleDate instance</td>
</tr>
<tr>
<td>GetDaysBetween</td>
<td>Calculates the number of days between the current OracleDate instance and an OracleDate structure</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToOracleTimeStamp</td>
<td>Converts the current OracleDate structure to an OracleTimeStamp structure</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleDate structure to a string</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.

**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**

```csharp
// 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.
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#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleDateSample
{
    static void Main(string[] args)
    {
        // Set the thread's DateFormat for the OracleDate constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "YYYY-MON-DD";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleDate from a string using the DateFormat specified.
        OracleDate date = new OracleDate("1999-DEC-01");

        // Set a different DateFormat for the thread
        info.DateFormat = "MM/DD/YYYY";
        OracleGlobalization.SetThreadInfo(info);
    }
}
OracleDate Constructors

// Print "12/01/1999"
Console.WriteLine(date.ToString());

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 6-2
- 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" on page 1-4
- 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).

**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**

```csharp
// 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" on page 1-4
- OracleDate Structure
- OracleDate Members
OracleDate Static Fields

The OracleDate static fields are listed in Table 9–21.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an OracleDate structure, which is December 31, 9999 23:59:59.</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an OracleDate structure, which is January 1, -4712 0:0:0.</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to the value of an OracleDate structure instance.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// C#
public static readonly OracleDate MaxValue;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members

MinValue

This static field represents the minimum valid date for an OracleDate structure, which is January 1, -4712.

Declaration

```csharp
// C#
public static readonly OracleDate MinValue;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members
Null

This static field represents a null value that can be assigned to the value of an OracleDate instance.

Decleration

```c#
public static readonly OracleDate Null;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members
OracleDate Static Methods

The `OracleDate` static methods are listed in Table 9–22.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code></td>
<td>Determines if two <code>OracleDate</code> values are equal (Overloaded)</td>
</tr>
<tr>
<td><code>GreaterThan</code></td>
<td>Determines if the first of two <code>OracleDate</code> values is greater than the second</td>
</tr>
<tr>
<td><code>GreaterThanOrEqual</code></td>
<td>Determines if the first of two <code>OracleDate</code> values is greater than or equal to the second</td>
</tr>
<tr>
<td><code>LessThan</code></td>
<td>Determines if the first of two <code>OracleDate</code> values is less than the second</td>
</tr>
<tr>
<td><code>LessThanOrEqual</code></td>
<td>Determines if the first of two <code>OracleDate</code> values is less than or equal to the second</td>
</tr>
<tr>
<td><code>NotEquals</code></td>
<td>Determines if two <code>OracleDate</code> values are not equal</td>
</tr>
<tr>
<td><code>GetSysDate</code></td>
<td>Returns an <code>OracleDate</code> structure that represents the current date and time</td>
</tr>
<tr>
<td><code>Parse</code></td>
<td>Returns an <code>OracleDate</code> structure and sets its value using a string</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleDate Structure`
- `OracleDate Members`

**Equals**

Overloads `Object`

This method determines if two `OracleDate` values are equal.

**Declaration**

```csharp
// 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.
OracleDate Static Methods

- 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" on page 1-4
- OracleDate Structure
- OracleDate Members

GreaterThan

This method determines if the first of two OracleDate values is greater than the second.

Declaration

```csharp
// 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" on page 1-4
- OracleDate Structure
- OracleDate Members

GreaterThanOrEqual

This method determines if the first of two OracleDate values is greater than or equal to the second.

Declaration

```csharp
// 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 `OracleDate` that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleDate Structure`
- `OracleDate Members`

LessThan
This method determines if the first of two `OracleDate` values is less than the second.

Declaration
```csharp
// 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 `OracleDate` that contain a null value are equal.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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.
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" on page 1-4
- 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" on page 1-4
- 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#
using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ParseSample
{
    static void Main(string[] args)
    {
        // Set the thread’s DateFormat for the OracleDate constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "YYYY-MON-DD";
        OracleGlobalization.SetThreadInfo(info);

        // Construct OracleDate from a string using the DateFormat specified
        OracleDate date = OracleDate.Parse("1999-DEC-01");

        // Set a different DateFormat on the thread for ToString()
        info.DateFormat = "MM-DD-YY";
        OracleGlobalization.SetThreadInfo(info);

        // Print "12-01-1999"
        Console.WriteLine(date.ToString());
    }
}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
- Oracle Database SQL Reference for further information on datetime format elements
OracleDate Static Operators

The OracleDate static operators are listed in Table 9–23.

Table 9–23  OracleDate Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator ==</td>
<td>Determines if two OracleDate values are the same</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleDate values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleDate values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if the two OracleDate values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleDate values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleDate values is less than or equal to the second</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members

operator ==

This method determines if two OracleDate values are the same.

Declaration

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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.
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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleDate Structure
- OracleDate Members
OracleDate Static Type Conversions

The OracleDate static type conversions are listed in Table 9–24.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator DateTime</td>
<td>Converts a structure to a DateTime structure</td>
</tr>
<tr>
<td>explicit operator OracleDate</td>
<td>Converts a structure to an OracleDate structure (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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.
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#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleDateSample
{
    static void Main(string[] args)
    {
        // Set the thread's DateFormat to a specific format
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = 'YYYY-MON-DD';
        OracleGlobalization.SetThreadInfo(info);
// Construct OracleDate from a string using the DateFormat specified
OracleDate date = (OracleDate)"1999-DEC-01";

// Set a different DateFormat on the thread for ToString()
info.DateFormat = "MON DD YY";
OracleGlobalization.SetThreadInfo(info);

// Prints 'DEC 01 99'
Console.WriteLine(date.ToString());
}
}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
OracleDate Properties

The OracleDate properties are listed in Table 9–25.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Gets an array of bytes that represents an Oracle DATE in Oracle internal format.</td>
</tr>
<tr>
<td>Day</td>
<td>Gets the day component of an OracleDate method.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value.</td>
</tr>
<tr>
<td>Hour</td>
<td>Gets the hour component of an OracleDate.</td>
</tr>
<tr>
<td>Minute</td>
<td>Gets the minute component of an OracleDate.</td>
</tr>
<tr>
<td>Month</td>
<td>Gets the month component of an OracleDate.</td>
</tr>
<tr>
<td>Second</td>
<td>Gets the second component of an OracleDate.</td>
</tr>
<tr>
<td>Value</td>
<td>Gets the date and time that is stored in the OracleDate structure.</td>
</tr>
<tr>
<td>Year</td>
<td>Gets the year component of an OracleDate.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members

BinData

This property gets an array of bytes that represents an Oracle DATE in Oracle internal format.

Declaration

```csharp
// C#
public byte[] BinData{get;}
```

Property Value

An array of bytes.

Exceptions

OracleNullValueException - OracleDate has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- `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" on page 1-4
- `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" on page 1-4
- `OracleDate Structure`
- `OracleDate Members`
OracleDate Properties

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" on page 1-4
- 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" on page 1-4
- 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.
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" on page 1-4
- `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" on page 1-4
- `OracleDate` Structure
- `OracleDate` Members
OracleDate Methods

The OracleDate methods are listed in Table 9–26.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleDate instance to an object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same date and time as the current OracleDate instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleDate instance</td>
</tr>
<tr>
<td>GetDaysBetween</td>
<td>Calculates the number of days between the current OracleDate instance and an OracleDate structure</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToOracleTimeStamp</td>
<td>Converts the current OracleDate structure to an OracleTimeStamp structure</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleDate structure to a string</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// 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**

ArgumentNullException - 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- `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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class ToStringSample
{
    static void Main(string[] args)
    {
        // Set the thread's DateFormat to a specific format
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.DateFormat = "YYYY-MON-DD";
        OracleGlobalization.SetThreadInfo(info);

        // Construct OracleDate from a string using the DateFormat specified
        OracleDate date = (OracleDate)"1999-DEC-01";

        // Set a different DateFormat on the thread for ToString()
        info.DateFormat = "YYYY/MM/DD";
        OracleGlobalization.SetThreadInfo(info);

        // Prints '1999/12/01'
Console.WriteLine(date.ToString());
}
}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
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#

using System;
using Oracle.DataAccess.Types;

class OracleDecimalSample
{
    static void Main(string[] args)
    {
        // Illustrates the range of OracleDecimal vs. .NET decimal
        OracleDecimal decimal1 = OracleDecimal.MinValue;
        OracleDecimal decimal2 = OracleDecimal.MaxValue;
        OracleDecimal decimal3 = new OracleDecimal(decimal.MinValue);
        OracleDecimal decimal4 = new OracleDecimal(decimal.MaxValue);

        // Print the ranges
        Console.WriteLine("OracleDecimal can range from\n{0}\nto\n{1}\n", decimal1, decimal2);
        Console.WriteLine(".NET decimal can range from\n{0}\nto\n{1}\n", decimal3, decimal4);
    }
}
Requirements
Namespace: `Oracle.DataAccess.Types`
Assembly: `Oracle.DataAccess.dll`

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 9–27.

Table 9–27 OracleDecimal Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleDecimal</td>
<td>Instantiates a new instance of OracleDecimal structure</td>
</tr>
<tr>
<td>Constructors</td>
<td>(Overloaded)</td>
</tr>
</tbody>
</table>

OracleDecimal Static Fields
The OracleDecimal static fields are listed in Table 9–28.

Table 9–28 OracleDecimal Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxPrecision</td>
<td>A constant representing the maximum precision, which is 38</td>
</tr>
<tr>
<td>MaxScale</td>
<td>A constant representing the maximum scale, which is 127</td>
</tr>
<tr>
<td>MaxValue</td>
<td>A constant representing the maximum value for this structure, which is 9.9...9 x 10^125</td>
</tr>
<tr>
<td>MinScale</td>
<td>A constant representing the minimum scale, which is -84</td>
</tr>
<tr>
<td>MinValue</td>
<td>A constant representing the minimum value for this structure, which is -1.0 x 10^130</td>
</tr>
<tr>
<td>NegativeOne</td>
<td>A constant representing the negative one value</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an OracleDecimal instance</td>
</tr>
<tr>
<td>One</td>
<td>A constant representing the positive one value</td>
</tr>
<tr>
<td>Pi</td>
<td>A constant representing the numeric Pi value</td>
</tr>
<tr>
<td>Zero</td>
<td>A constant representing the zero value</td>
</tr>
</tbody>
</table>

OracleDecimal Static (Comparison) Methods
The OracleDecimal static (comparison) methods are listed in Table 9–29.

Table 9–29 OracleDecimal Static (Comparison) Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleDecimal values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleDecimal values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleDecimal values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleDecimal values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleDecimal values is less than or equal to the second</td>
</tr>
</tbody>
</table>
OracleDecimal Members

Table 9–29  (Cont.) OracleDecimal Static (Comparison) Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleDecimal values are not equal</td>
</tr>
</tbody>
</table>

OracleDecimal Static (Manipulation) Methods

The OracleDecimal static (manipulation) methods are listed in Table 9–30.

Table 9–30  OracleDecimal Static (Manipulation) Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>Returns the absolute value of an OracleDecimal</td>
</tr>
<tr>
<td>Add</td>
<td>Adds two OracleDecimal structures</td>
</tr>
<tr>
<td>AdjustScale</td>
<td>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</td>
</tr>
<tr>
<td>Ceiling</td>
<td>Returns a new OracleDecimal structure with its value set to the ceiling of an OracleDecimal structure</td>
</tr>
<tr>
<td>ConvertToPrecScale</td>
<td>Returns a new OracleDecimal structure with a new precision and scale</td>
</tr>
<tr>
<td>Divide</td>
<td>Divides one OracleDecimal value by another</td>
</tr>
<tr>
<td>Floor</td>
<td>Returns a new OracleDecimal structure with its value set to the floor of an OracleDecimal structure</td>
</tr>
<tr>
<td>Max</td>
<td>Returns the maximum value of the two supplied OracleDecimal structures</td>
</tr>
<tr>
<td>Min</td>
<td>Returns the minimum value of the two supplied OracleDecimal structures</td>
</tr>
<tr>
<td>Mod</td>
<td>Returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures</td>
</tr>
<tr>
<td>Multiply</td>
<td>Returns a new OracleDecimal structure with its value set to the result of multiplying two OracleDecimal structures</td>
</tr>
<tr>
<td>Negate</td>
<td>Returns a new OracleDecimal structure with its value set to the negation of the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Parse</td>
<td>Converts a string to an OracleDecimal</td>
</tr>
<tr>
<td>Round</td>
<td>Returns a new OracleDecimal structure with its value set to that of the supplied OracleDecimal structure and rounded off to the specified place</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new OracleDecimal structure with a new specified precision.</td>
</tr>
<tr>
<td>Shift</td>
<td>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</td>
</tr>
<tr>
<td>Sign</td>
<td>Determines the sign of an OracleDecimal structure</td>
</tr>
<tr>
<td>Sqrt</td>
<td>Returns a new OracleDecimal structure with its value set to the square root of the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Subtract</td>
<td>Returns a new OracleDecimal structure with its value set to result of subtracting one OracleDecimal structure from another</td>
</tr>
<tr>
<td>Truncate</td>
<td>Truncates the OracleDecimal at a specified position</td>
</tr>
</tbody>
</table>
OracleDecimal Static (Logarithmic) Methods

The OracleDecimal static (logarithmic) methods are listed in Table 9–31.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp</td>
<td>Returns a new OracleDecimal structure with its value set to e raised to the supplied power</td>
</tr>
<tr>
<td>Log</td>
<td>Returns the supplied OracleDecimal structure with its value set to the logarithm of the supplied OracleDecimal structure (Overloaded)</td>
</tr>
<tr>
<td>Pow</td>
<td>Returns a new OracleDecimal structure with its value set to the supplied OracleDecimal structure raised to the supplied power (Overloaded)</td>
</tr>
</tbody>
</table>

OracleDecimal Static (Trigonometric) Methods

The OracleDecimal static (trigonometric) methods are listed in Table 9–32.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acos</td>
<td>Returns an angle in radian whose cosine is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Asin</td>
<td>Returns an angle in radian whose sine is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Atan</td>
<td>Returns an angle in radian whose tangent is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Atan2</td>
<td>Returns an angle in radian whose tangent is the quotient of the two supplied OracleDecimal structures</td>
</tr>
<tr>
<td>Cos</td>
<td>Returns the cosine of the supplied angle in radian</td>
</tr>
<tr>
<td>Sin</td>
<td>Returns the sine of the supplied angle in radian</td>
</tr>
<tr>
<td>Tan</td>
<td>Returns the tangent of the supplied angle in radian</td>
</tr>
<tr>
<td>Cosh</td>
<td>Returns the hyperbolic cosine of the supplied angle in radian</td>
</tr>
<tr>
<td>Sinh</td>
<td>Returns the hyperbolic sine of the supplied angle in radian</td>
</tr>
<tr>
<td>Tanh</td>
<td>Returns the hyperbolic tangent of the supplied angle in radian</td>
</tr>
</tbody>
</table>

OracleDecimal Static (Comparison) Operators

The OracleDecimal static (comparison) operators are listed in Table 9–33.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+</code></td>
<td>Adds two OracleDecimal values</td>
</tr>
<tr>
<td><code>/</code></td>
<td>Divides one OracleDecimal value by another</td>
</tr>
<tr>
<td><code>==</code></td>
<td>Determines if the two OracleDecimal values are equal</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Determines if the first of two OracleDecimal values is greater than the second</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Determines if the first of two OracleDecimal values is greater than or equal to the second</td>
</tr>
</tbody>
</table>
OracleDecimal Members

OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)
The OracleDecimal static operators (Conversion from .NET Type to OracleDecimal) are listed in Table 9–34.

### Table 9–34  OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>implicit operator OracleDecimal</td>
<td>Converts an instance value to an OracleDecimal structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator OracleDecimal</td>
<td>Converts an instance value to an OracleDecimal structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)
The OracleDecimal static operators (Conversion from OracleDecimal to .NET) are listed in Table 9–35.

### Table 9–35  OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator byte</td>
<td>Returns the byte representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator decimal</td>
<td>Returns the decimal representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator double</td>
<td>Returns the double representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator short</td>
<td>Returns the short representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator int</td>
<td>Returns the int representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator long</td>
<td>Returns the long representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator float</td>
<td>Returns the float representation of the OracleDecimal value</td>
</tr>
</tbody>
</table>

OracleDecimal Properties
The OracleDecimal properties are listed in Table 9–36.
**OracleDecimal Properties**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns a byte array that represents the Oracle NUMBER in Oracle internal format</td>
</tr>
<tr>
<td>Format</td>
<td>Specifies the format for ToString()</td>
</tr>
<tr>
<td>IsInt</td>
<td>Indicates whether the current instance is an integer</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>IsPositive</td>
<td>Indicates whether the current instance is greater than 0</td>
</tr>
<tr>
<td>IsZero</td>
<td>Indicates whether the current instance has a zero value</td>
</tr>
<tr>
<td>Value</td>
<td>Returns a decimal value</td>
</tr>
</tbody>
</table>

**OracleDecimal Instance Methods**

The OracleDecimal instance methods are listed in Table 9–37.

**Table 9–37  OracleDecimal Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current instance to the supplied object and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object is an instance of OracleDecimal, and whether the value of the object is equal to the current instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the current instance</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToByte</td>
<td>Returns the byte representation of the current instance</td>
</tr>
<tr>
<td>ToDouble</td>
<td>Returns the double representation of the current instance</td>
</tr>
<tr>
<td>ToInt16</td>
<td>Returns the Int16 representation of the current instance</td>
</tr>
<tr>
<td>ToInt32</td>
<td>Returns the Int32 representation of the current instance</td>
</tr>
<tr>
<td>ToInt64</td>
<td>Returns the Int64 representation of the current instance</td>
</tr>
<tr>
<td>ToSingle</td>
<td>Returns the Single representation of the current instance</td>
</tr>
<tr>
<td>ToString</td>
<td>Overloads Object.ToString() Returns the string representation of the current instance</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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**

```csharp
// 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.

---

**OracleDecimal(decimal)**

This constructor creates a new instance of the `OracleDecimal` structure and sets its value to the supplied `Decimal` value.

Declaration

```csharp
// C#
public OracleDecimal(decimal decX);
```

Parameters

- **decX**
  
  The provided `Decimal` value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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

```csharp
// 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`.

---

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleDecimal Members`
- `OracleDecimal Structure`
OracleDecimal Constructors

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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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.
OracleDecimal Constructors

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleDecimalSample
{
    static void Main(string[] args)
    {


// Set the nls parameters related to currency
OracleGlobalization info = OracleGlobalization.GetClientInfo();
info.Currency = "$";
info.NumericCharacters = ",.\";
OracleGlobalization.SetThreadInfo(info);

// Construct an OracleDecimal using a valid numeric format
OracleDecimal dec = new OracleDecimal("$2,222.22","L9G999D99");

// Print "$2,222.22"
Console.WriteLine(dec.ToString());
}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
OracleDecimal Static Fields

The OracleDecimal static fields are listed in Table 9–38.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxPrecision</td>
<td>A constant representing the maximum precision, which is 38</td>
</tr>
<tr>
<td>MaxScale</td>
<td>A constant representing the maximum scale, which is 127</td>
</tr>
<tr>
<td>MaxValue</td>
<td>A constant representing the maximum value for this structure, which is</td>
</tr>
<tr>
<td></td>
<td>$9.9...9 \times 10^{125}$</td>
</tr>
<tr>
<td>MinScale</td>
<td>A constant representing the minimum scale, which is -84</td>
</tr>
<tr>
<td>MinValue</td>
<td>A constant representing the minimum value for this structure, which is</td>
</tr>
<tr>
<td></td>
<td>$-1.0 \times 10^{130}$</td>
</tr>
<tr>
<td>NegativeOne</td>
<td>A constant representing the negative one value</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an OracleDecimal instance</td>
</tr>
<tr>
<td>One</td>
<td>A constant representing the positive one value</td>
</tr>
<tr>
<td>Pi</td>
<td>A constant representing the numeric Pi value</td>
</tr>
<tr>
<td>Zero</td>
<td>A constant representing the zero value</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

MaxPrecision

This static field represents the maximum precision, which is 38.

Declaration

```csharp
// C#
public static readonly byte MaxPrecision;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

MaxScale

This static field a constant representing the maximum scale, which is 127.

Declaration

```csharp
// C#
public static readonly byte MaxScale;
```
MaxValue

This static field indicates a constant representing the maximum value for this structure, which is $9.9\ldots9 \times 10^{125}$ (38 nines followed by 88 zeroes).

Declaration

```csharp
// C#
public static readonly OracleDecimal MaxValue;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

MinScale

This static field indicates a constant representing the maximum scale, which is -84.

Declaration

```csharp
// C#
public static readonly int MinScale;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

MinValue

This static field indicates a constant representing the minimum value for this structure, which is $-1.0 \times 10^{150}$.

Declaration

```csharp
// C#
public static readonly OracleDecimal MinValue;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

NegativeOne

This static field indicates a constant representing the negative one value.
OracleDecimal Static Fields

Declaration

// C#
public static readonly OracleDecimal NegativeOne;

See Also:

- "Oracle.DataAccess.Types Namespace” on page 1-4
- 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” on page 1-4
- 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” on page 1-4
- 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” on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Comparison) Methods

The OracleDecimal static (comparison) methods are listed in Table 9–39.

Table 9–39  OracleDecimal Static (Comparison) Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleDecimal values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleDecimal values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleDecimal values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleDecimal values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleDecimal values is less than or equal to the second.</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleDecimal values are not equal</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

Equals

This method determines if two OracleDecimal values are equal.

Declaration

```csharp
// 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.
GreaterThan

This method determines if the first of two OracleDecimal values is greater than the second.

**Declaration**

```csharp
// 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

GreaterThanOrEqual

This method determines if the first of two OracleDecimal values is greater than or equal to the second.

**Declaration**

```csharp
// C#
public static bool GreaterThanOrEqual(OracleDecimal value1, OracleDecimal value2);
```

**Parameters**

- **value1**
  - First OracleDecimal.
- **value2**
  - Second OracleDecimal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
■ OracleDecimal Members
■ OracleDecimal Structure
OracleDecimal Static (Manipulation) Methods

The OracleDecimal static (manipulation) methods are listed in Table 9–40.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>Returns the absolute value of an OracleDecimal</td>
</tr>
<tr>
<td>Add</td>
<td>Adds two OracleDecimal structures</td>
</tr>
<tr>
<td>AdjustScale</td>
<td>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</td>
</tr>
<tr>
<td>Ceiling</td>
<td>Returns a new OracleDecimal structure with its value set to the ceiling of an OracleDecimal structure</td>
</tr>
<tr>
<td>ConvertToPrecScale</td>
<td>Returns a new OracleDecimal structure with a new precision and scale</td>
</tr>
<tr>
<td>Divide</td>
<td>Divides one OracleDecimal value by another</td>
</tr>
<tr>
<td>Floor</td>
<td>Returns a new OracleDecimal structure with its value set to the floor of an OracleDecimal structure</td>
</tr>
<tr>
<td>Max</td>
<td>Returns the maximum value of the two supplied OracleDecimal structures</td>
</tr>
<tr>
<td>Min</td>
<td>Returns the minimum value of the two supplied OracleDecimal structures</td>
</tr>
<tr>
<td>Mod</td>
<td>Returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures</td>
</tr>
<tr>
<td>Multiply</td>
<td>Returns a new OracleDecimal structure with its value set to the result of multiplying two OracleDecimal structures</td>
</tr>
<tr>
<td>Negate</td>
<td>Returns a new OracleDecimal structure with its value set to the negation of the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Parse</td>
<td>Converts a string to an OracleDecimal</td>
</tr>
<tr>
<td>Round</td>
<td>Returns a new OracleDecimal structure with its value set to that of the supplied OracleDecimal structure and rounded off to the specified place</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new OracleDecimal structure with a new specified precision.</td>
</tr>
<tr>
<td>Shift</td>
<td>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</td>
</tr>
<tr>
<td>Sign</td>
<td>Determines the sign of an OracleDecimal structure</td>
</tr>
<tr>
<td>Sqrt</td>
<td>Returns a new OracleDecimal structure with its value set to the square root of the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Subtract</td>
<td>Returns a new OracleDecimal structure with its value set to result of subtracting one OracleDecimal structure from another</td>
</tr>
<tr>
<td>Truncate</td>
<td>Truncates the OracleDecimal at a specified position</td>
</tr>
</tbody>
</table>
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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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#
using System;
using Oracle.DataAccess.Types;

class AdjustScaleSample
{
  static void Main(string[] args)
  {
    OracleDecimal dec1 = new OracleDecimal(5.555);

    // Adjust Scale to 2 with rounding off
    OracleDecimal dec2 = OracleDecimal.AdjustScale(dec1, 2, true);

    // Prints 5.56
    Console.WriteLine(dec2.ToString());

    // Adjust Scale to 2 with truncation
    OracleDecimal dec3 = OracleDecimal.AdjustScale(dec1, 2, false);
// Prints 5.55
Console.WriteLine(dec3.ToString());
}
}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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**

```csharp
using System;
using Oracle.DataAccess.Types;

class ConvertToPrecScaleSample
{
    static void Main(string[] args)
    {
        OracleDecimal dec1 = new OracleDecimal(555.6666);

        // Set the precision of od to 5 and scale to 2
        OracleDecimal dec2 = OracleDecimal.ConvertToPrecScale(dec1, 5, 2);

        // Prints 555.67
        Console.WriteLine(dec2.ToString());

        // Set the precision of od to 3 and scale to 0
        OracleDecimal dec3 = OracleDecimal.ConvertToPrecScale(dec1, 3, 0);

        // Prints 556
        Console.WriteLine(dec3.ToString());
    }
}
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleDecimal Members`
- `OracleDecimal Structure`

**Divide**

This method divides one `OracleDecimal` value by another.

**Declaration**

```csharp
// 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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

**Max**

This method returns the maximum value of the two supplied OracleDecimal structures.

**Declaration**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- `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**

```csharp
// 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" on page 1-4
- `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**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

SetPrecision

This method returns a new OracleDecimal structure with a new specified precision.

Declaration

```csharp
// 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

```csharp
// C#
using System;
using Oracle.DataAccess.Types;

class SetPrecisionSample
```
{ static void Main(string[] args)
{
 OracleDecimal dec1 = new OracleDecimal(555.6666);

 // Set the precision of dec1 to 3
 OracleDecimal dec2 = OracleDecimal.SetPrecision(dec1, 3);

 // Prints 556
 Console.WriteLine(dec2.ToString());

 // Set the precision of dec1 to 4
 OracleDecimal dec3 = OracleDecimal.SetPrecision(dec1, 4);

 // Prints 555.7
 Console.WriteLine(dec3.ToString());
}
}

See Also:
- "Oracle.DataAccess.Types Namespace” on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
Remarks
If either argument has a null value, the returned OracleDecimal has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Logarithmic) Methods

The OracleDecimal static (logarithmic) methods are listed in Table 9–41.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp</td>
<td>Returns a new OracleDecimal structure with its value set to e raised to the supplied power</td>
</tr>
<tr>
<td>Log</td>
<td>Returns the supplied OracleDecimal structure with its value set to the logarithm of the supplied OracleDecimal structure (Overloaded)</td>
</tr>
<tr>
<td>Pow</td>
<td>Returns a new OracleDecimal structure with its value set to the supplied OracleDecimal structure raised to the supplied power (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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

```csharp
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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Trigonometric) Methods

The OracleDecimal static (trigonometric) methods are listed in Table 9–42.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acos</td>
<td>Returns an angle in radian whose cosine is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Asin</td>
<td>Returns an angle in radian whose sine is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Atan</td>
<td>Returns an angle in radian whose tangent is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Atan2</td>
<td>Returns an angle in radian whose tangent is the quotient of the two supplied OracleDecimal structures</td>
</tr>
<tr>
<td>Cos</td>
<td>Returns the cosine of the supplied angle in radian</td>
</tr>
<tr>
<td>Sin</td>
<td>Returns the sine of the supplied angle in radian</td>
</tr>
<tr>
<td>Tan</td>
<td>Returns the tangent of the supplied angle in radian</td>
</tr>
<tr>
<td>Cosh</td>
<td>Returns the hyperbolic cosine of the supplied angle in radian</td>
</tr>
<tr>
<td>Sinh</td>
<td>Returns the hyperbolic sine of the supplied angle in radian</td>
</tr>
<tr>
<td>Tanh</td>
<td>Returns the hyperbolic tangent of the supplied angle in radian</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

Acos

This method returns an angle in radian whose cosine is the supplied OracleDecimal structure.

Declaration

```csharp
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.
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" on page 1-4
- 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" on page 1-4
- 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” on page 1-4
■ 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

ArgumentException - 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- `OracleDecimal` Members
- `OracleDecimal` Structure

### Cosh

This method returns the hyperbolic cosine of the supplied angle in radian.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleDecimal` Members
- `OracleDecimal` Structure

### Sinh

This method returns the hyperbolic sine of the supplied angle in radian.

**Declaration**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Comparison) Operators

The `OracleDecimal` static (comparison) operators are listed in Table 9–43.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>operator +</code></td>
<td>Adds two <code>OracleDecimal</code> values</td>
</tr>
<tr>
<td><code>operator /</code></td>
<td>Divides one <code>OracleDecimal</code> value by another</td>
</tr>
<tr>
<td><code>operator ==</code></td>
<td>Determines if the two <code>OracleDecimal</code> values are equal</td>
</tr>
<tr>
<td><code>operator &gt;</code></td>
<td>Determines if the first of two <code>OracleDecimal</code> values is greater than the second</td>
</tr>
<tr>
<td><code>operator &gt;=</code></td>
<td>Determines if the first of two <code>OracleDecimal</code> values is greater than or equal to the second</td>
</tr>
<tr>
<td><code>operator !=</code></td>
<td>Determines if the two <code>OracleDecimal</code> values are not equal</td>
</tr>
<tr>
<td><code>operator &lt;</code></td>
<td>Determines if the first of two <code>OracleDecimal</code> values is less than the second</td>
</tr>
<tr>
<td><code>operator &lt;=</code></td>
<td>Determines if the first of two <code>OracleDecimal</code> values is less than or equal to the second</td>
</tr>
<tr>
<td><code>operator *</code></td>
<td>Multiplies two <code>OracleDecimal</code> structures</td>
</tr>
<tr>
<td><code>operator -</code></td>
<td>Subtracts one <code>OracleDecimal</code> structure from another</td>
</tr>
<tr>
<td><code>operator -</code></td>
<td>Negates an <code>OracleDecimal</code> structure</td>
</tr>
<tr>
<td><code>operator%</code></td>
<td>Returns a new <code>OracleDecimal</code> structure with its value set to the modulus of two <code>OracleDecimal</code> structures.</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

**operator +**

This method adds two `OracleDecimal` values.

**Declaration**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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 `OracleDecimal` values that contain a null value are equal.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleDecimal Members`
- `OracleDecimal Structure`

---

**operator >**

This method determines if the first of two `OracleDecimal` values is greater than the second.

**Declaration**

```csharp
// 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 `OracleDecimal` values that contain a null value are equal.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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" on page 1-4
- 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.

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.

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 `OracleDecimal`s that contain a null value are equal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleDecimal Members`
- `OracleDecimal Structure`

**operator ***

This method multiplies two `OracleDecimal` structures.

**Declaration**

```csharp
// 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" on page 1-4
- `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" on page 1-4
■ 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" on page 1-4
■ 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" on page 1-4
- 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 9–44.

Table 9–44   OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>implicit operator OracleDecimal</td>
<td>Converts an instance value to an OracleDecimal structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator OracleDecimal</td>
<td>Converts an instance value to an OracleDecimal structure (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

**explicit operator OracleDecimal(double)**

This method returns the OracleDecimal representation of a double.

**Declaration**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

The OracleDecimal static operators (Conversion from OracleDecimal to .NET) are listed in Table 9–45.

Table 9–45  OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator byte</td>
<td>Returns the byte representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator decimal</td>
<td>Returns the decimal representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator double</td>
<td>Returns the double representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator short</td>
<td>Returns the short representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator int</td>
<td>Returns the int representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator long</td>
<td>Returns the long representation of the OracleDecimal value</td>
</tr>
<tr>
<td>explicit operator float</td>
<td>Returns the float representation of the OracleDecimal value</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
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" on page 1-4
- 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.
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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Structure

explicit operator long

This method returns the long representation of the OracleDecimal value.

Declaration

```csharp
// 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

explicit operator float

This method returns the float representation of the OracleDecimal value.

Declaration

```csharp
// 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Properties

The OracleDecimal properties are listed in Table 9–46.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns a byte array that represents the Oracle NUMBER in Oracle internal format</td>
</tr>
<tr>
<td>Format</td>
<td>Specifies the format for ToString()</td>
</tr>
<tr>
<td>IsInt</td>
<td>Indicates whether the current instance is an integer</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>IsPositive</td>
<td>Indicates whether the current instance is greater than 0</td>
</tr>
<tr>
<td>IsZero</td>
<td>Indicates whether the current instance has a zero value</td>
</tr>
<tr>
<td>Value</td>
<td>Returns a decimal value</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- `OracleDecimal Members`
- `OracleDecimal Structure`
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

**IsInt**
This property indicates whether the current instance is an integer value.

**Declaration**
```csharp
// 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" on page 1-4
- `OracleDecimal Members`
- `OracleDecimal Structure`

**IsNull**
This property indicates whether the current instance has a null value.

**Declaration**
```csharp
// C#
```

9-130  Oracle Data Provider for .NET Developer's Guide
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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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 9-65 for further information.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Instance Methods

The OracleDecimal instance methods are listed in Table 9–47.

**Table 9–47  OracleDecimal Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current instance to the supplied object and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object is an instance of OracleDecimal, and whether the value of the object is equal to the current instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the current instance</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToByte</td>
<td>Returns the byte representation of the current instance</td>
</tr>
<tr>
<td>ToDouble</td>
<td>Returns the double representation of the current instance</td>
</tr>
<tr>
<td>ToInt16</td>
<td>Returns the Int16 representation of the current instance</td>
</tr>
<tr>
<td>ToInt32</td>
<td>Returns the Int32 representation of the current instance</td>
</tr>
<tr>
<td>ToInt64</td>
<td>Returns the Int64 representation of the current instance</td>
</tr>
<tr>
<td>ToSingle</td>
<td>Returns the Single representation of the current instance</td>
</tr>
<tr>
<td>ToString</td>
<td>Overloads Object.ToString() Returns the string representation of the current instance</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// 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`.
OracleDecimal Instance Methods

**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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

**Equals**

Overides 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.
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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure

ToDouble

This method returns the double representation of the current instance.

Declaration

// C#
public double ToDouble();
OracleDecimal Instance Methods

**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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
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 terms 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#
using System;
using Oracle.DataAccess.Types;

class OracleIntervalDSSample
{
    static void Main()
    {
        OracleIntervalDS iDSMax = OracleIntervalDS.MaxValue;
        double totalDays = iDSMax.TotalDays;
        totalDays -= 1;
        OracleIntervalDS iDSMax_1 = new OracleIntervalDS(totalDays);
        
        // Calculate the difference
        OracleIntervalDS iDSDiff = iDSMax - iDSMax_1;

        // Prints "iDSDiff.ToString() = +00000000 23:59:59.999999999"
        Console.WriteLine("iDSDiff.ToString() = " + iDSDiff.ToString());
    }
}

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 9–48.

Table 9–48  OracleIntervalDS Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleIntervalDS Constructors</td>
<td>Instantiates a new instance of OracleIntervalDS structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleIntervalDS Static Fields
The OracleIntervalDS static fields are listed in Table 9–49.

Table 9–49  OracleIntervalDS Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid time interval for an OracleIntervalDS structure</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid time interval for an OracleIntervalDS structure</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an OracleIntervalDS instance</td>
</tr>
<tr>
<td>Zero</td>
<td>Represents a zero value for an OracleIntervalDS structure</td>
</tr>
</tbody>
</table>

OracleIntervalDS Static Methods
The OracleIntervalDS static methods are listed in Table 9–50.

Table 9–50  OracleIntervalDS Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether two OracleIntervalDS values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether one OracleIntervalDS value is greater than another</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether one OracleIntervalDS value is greater than or equal to another</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether one OracleIntervalDS value is less than another</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether one OracleIntervalDS value is less than or equal to another</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines whether two OracleIntervalDS values are not equal</td>
</tr>
<tr>
<td>Parse</td>
<td>Returns an OracleIntervalDS structure and sets its value for time interval using a string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleIntervalDS with the specified day precision and fractional second precision</td>
</tr>
</tbody>
</table>
OracleIntervalDS Members

OracleIntervalDS Static Operators
The OracleIntervalDS static operators are listed in Table 9–51.

Table 9–51  OracleIntervalDS Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds two OracleIntervalDS values</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines whether two OracleIntervalDS values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines whether one OracleIntervalDS value is greater than another</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines whether one OracleIntervalDS value is greater than or equal to another</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines whether two OracleIntervalDS values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines whether one OracleIntervalDS value is less than another</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines whether one OracleIntervalDS value is less than or equal to another</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts one OracleIntervalDS value from another</td>
</tr>
<tr>
<td>operator -</td>
<td>Negates an OracleIntervalDS structure</td>
</tr>
<tr>
<td>operator *</td>
<td>Multiplies an OracleIntervalDS value by a number</td>
</tr>
<tr>
<td>operator /</td>
<td>Divides an OracleIntervalDS value by a number</td>
</tr>
</tbody>
</table>

OracleIntervalDS Type Conversions
The OracleIntervalDS type conversions are listed in Table 9–52.

Table 9–52  OracleIntervalDS Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator TimeSpan</td>
<td>Converts an OracleIntervalDS structure to a TimeSpan structure</td>
</tr>
<tr>
<td>explicit operator OracleIntervalDS</td>
<td>Converts a string to an OracleIntervalDS structure</td>
</tr>
<tr>
<td>implicit operator OracleIntervalDS</td>
<td>Converts a TimeSpan structure to an OracleIntervalDS structure</td>
</tr>
</tbody>
</table>

OracleIntervalDS Properties
The OracleIntervalDS properties are listed in Table 9–53.

Table 9–53  OracleIntervalDS Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents the Oracle INTERVAL DAY TO SECOND in Oracle internal format</td>
</tr>
<tr>
<td>Days</td>
<td>Gets the days component of an OracleIntervalDS</td>
</tr>
</tbody>
</table>
OracleIntervalDS Methods

The `OracleIntervalDS` methods are listed in Table 9–54.

Table 9–54  OracleIntervalDS Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current <code>OracleIntervalDS</code> instance to an object, and returns</td>
</tr>
<tr>
<td></td>
<td>an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object has the same time interval as the</td>
</tr>
<tr>
<td></td>
<td>current instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the <code>OracleIntervalDS</code> instance</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from <code>Object</code></td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current <code>OracleIntervalDS</code> structure to a string</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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)**
  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" on page 1-4
- 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**

```csharp
// C#
public OracleIntervalDS(TimeSpan ts);
```

**Parameters**

- **ts**
  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.

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:SSxxFF 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" on page 1-4
- 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.
OracleIntervalDS Constructors

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" on page 1-4
- 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" on page 1-4
- 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**

- 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" on page 1-4
- 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS Static Fields

The OracleIntervalDS static fields are listed in Table 9–55.

Table 9–55  OracleIntervalDS Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid time interval for an OracleIntervalDS structure</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid time interval for an OracleIntervalDS structure</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an OracleIntervalDS instance</td>
</tr>
<tr>
<td>Zero</td>
<td>Represents a zero value for an OracleIntervalDS structure</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

MinValue

This static field represents the minimum value for an OracleIntervalDS structure.

Declaration

// C#
public static readonly OracleIntervalDS MinValue;
OracleIntervalDS Static Fields

Remarks
Minimum values:
- Day: -999999999
- hour: -23
- minute: -59
- second: -59
- nanosecond: -999999999

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS Static Methods

The OracleIntervalDS static methods are listed in Table 9–56.

Table 9–56  OracleIntervalDS Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether two OracleIntervalDS values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether one OracleIntervalDS value is greater than another</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether one OracleIntervalDS value is greater than or equal to another</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether one OracleIntervalDS value is less than another</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether one OracleIntervalDS value is less than or equal to another</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines whether two OracleIntervalDS values are not equal</td>
</tr>
<tr>
<td>Parse</td>
<td>Returns an OracleIntervalDS structure and sets its value for time interval using a string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleIntervalDS with the specified day precision and fractional second precision</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

Equals

This static method determines whether two OracleIntervalDS values are equal.

Declaration

```csharp
// 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 OracleIntervalDSs that contain a null value are equal.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleIntervalDSs that contain a null value are equal.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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);
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members
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* that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- *OracleIntervalDS Structure*
- *OracleIntervalDS Members*

**LessThan**

This static method determines whether the first of two *OracleIntervalDS* values is less than the second.

Declaration

```csharp
// 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* that contain a null value are equal.
LessThanOrEqual

This static method determines whether the first of two `OracleIntervalDS` values is less than or equal to the second.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleIntervalDS` Structure
- `OracleIntervalDS` Members

NotEquals

This static method determines whether two `OracleIntervalDS` values are not equal.

**Declaration**

```csharp
// C#
public static bool NotEquals(OracleIntervalDS val1, OracleIntervalDS val2);
```

**Parameters**

- `val1`  
  First `OracleIntervalDS`.
- `val2`  
  Second `OracleIntervalDS`.
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 `OracleIntervalDS`s that contain a null value are equal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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.
SetPrecision

This static method returns a new instance of an `OracleIntervalDS` with the specified day precision and fractional second precision.

**Declaration**

```csharp
// 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.
OracleIntervalDS Static Operators

The OracleIntervalDS static operators are listed in Table 9–57.

Table 9–57  OracleIntervalDS Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds two OracleIntervalDS values</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines whether two OracleIntervalDS values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines whether one OracleIntervalDS value is greater than another</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines whether one OracleIntervalDS value is greater than or equal to another</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines whether two OracleIntervalDS values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines whether one OracleIntervalDS value is less than another</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines whether one OracleIntervalDS value is less than or equal to another</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts one OracleIntervalDS value from another</td>
</tr>
<tr>
<td>operator -</td>
<td>Negates an OracleIntervalDS structure</td>
</tr>
<tr>
<td>operator *</td>
<td>Multiplies an OracleIntervalDS value by a number</td>
</tr>
<tr>
<td>operator /</td>
<td>Divides an OracleIntervalDS value by a number</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
OracleIntervalDS Static Operators

Remarks
If either argument has a null value, the returned OracleIntervalDS structure has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleIntervalDSs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- `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**

```csharp
// 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator !=

This static operator determines if the two OracleIntervalDS values are not equal.

Declaration

```csharp
// 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 OracleIntervalDSs that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator <

This static operator determines if the first of two OracleIntervalDS values is less than the second.

Declaration

```csharp
// C#
public static bool operator < (OracleIntervalDS val1,
   OracleIntervalDS val2);
```

Parameters

- **val1**
  First 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.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleIntervalDSs that contain a null value are equal.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator -

This static operator subtracts one OracleIntervalDS structure from another.

Declaration

```csharp
// 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator -

This static operator negates the supplied OracleIntervalDS structure.

Declaration

```csharp
// 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" on page 1-4
- `OracleIntervalDS` Structure
- `OracleIntervalDS` Members

**operator ***

This static operator multiplies an `OracleIntervalDS` value by a number.

Declaration

```csharp
// 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" on page 1-4
- `OracleIntervalDS` Structure
- `OracleIntervalDS` Members

**operator /**

This static operator divides an `OracleIntervalDS` value by a number.

Declaration

```csharp
// 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS Type Conversions

The OracleIntervalDS type conversions are listed in Table 9–58.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator TimeSpan</td>
<td>Converts an OracleIntervalDS structure to a TimeSpan structure</td>
</tr>
<tr>
<td>explicit operator OracleIntervalDS</td>
<td>Converts a string to an OracleIntervalDS structure</td>
</tr>
<tr>
<td>implicit operator OracleIntervalDS</td>
<td>Converts a TimeSpan structure to an OracleIntervalDS structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

explicit operator TimeSpan

This type conversion operator converts an OracleIntervalDS structure to a TimeSpan structure.

Declaration

```csharp
// 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" on page 1-4
- 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:SS:xxFF 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" on page 1-4
- 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members
### OracleIntervalDS Properties

The `OracleIntervalDS` properties are listed in Table 9–59.

#### Table 9–59  OracleIntervalDS Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents the Oracle <code>INTERVAL DAY TO SECOND</code> in Oracle internal format</td>
</tr>
<tr>
<td>Days</td>
<td>Gets the days component of an <code>OracleIntervalDS</code></td>
</tr>
<tr>
<td>Hours</td>
<td>Gets the hours component of an <code>OracleIntervalDS</code></td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>Milliseconds</td>
<td>Gets the milliseconds component of an <code>OracleIntervalDS</code></td>
</tr>
<tr>
<td>Minutes</td>
<td>Gets the minutes component of an <code>OracleIntervalDS</code></td>
</tr>
<tr>
<td>Nanoseconds</td>
<td>Gets the nanoseconds component of an <code>OracleIntervalDS</code></td>
</tr>
<tr>
<td>Seconds</td>
<td>Gets the seconds component of an <code>OracleIntervalDS</code></td>
</tr>
<tr>
<td>TotalDays</td>
<td>Returns the total number, in days, that represent the time period in the <code>OracleIntervalDS</code> structure</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the time interval that is stored in the <code>OracleIntervalDS</code> structure</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

**Milliseconds**
This property gets the milliseconds component of an OracleIntervalDS.

**Declaration**
```csharp
// 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members

**Minutes**
This property gets the minutes component of an OracleIntervalDS.

**Declaration**
```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS Methods

The OracleIntervalDS methods are listed in Table 9–60.

### Table 9–60 OracleIntervalDS Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleIntervalDS instance to an object, and returns an integer that represents their relative values.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object has the same time interval as the current instance (Overloaded).</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleIntervalDS instance.</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleIntervalDS structure to a string.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// 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 OracleIntervalDSs. 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 OracleIntervalDSs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleIntervalDSs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleIntervalDS Structure
- OracleIntervalDS Members
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#

using System;
using Oracle.DataAccess.Types;

class OracleIntervalYMSample
{
    static void Main()
    {
        OracleIntervalYM iYMMax = OracleIntervalYM.MaxValue;
        double totalYears = iYMMax.TotalYears;

        totalYears -= 1;
        OracleIntervalYM iYMMax_1 = new OracleIntervalYM(totalYears);

        // Calculate the difference
        OracleIntervalYM iYMDiff = iYMMax - iYMMax_1;

        // Prints "iYMDiff.ToString() = +000000001-00"
        Console.WriteLine("iYMDiff.ToString() = " + iYMDiff.ToString());
    }
}

**Requirements**

Namespace: Oracle.DataAccess.Types

Assembly: Oracle.DataAccess.dll
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 9–61

Table 9–61  OracleIntervalYM Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleIntervalYM Constructors</td>
<td>Instantiates a new instance of OracleIntervalYM structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleIntervalYM Static Fields
The OracleIntervalYM static fields are listed in Table 9–62.

Table 9–62  OracleIntervalYM Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum value for an OracleIntervalYM structure</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum value for an OracleIntervalYM structure</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an OracleIntervalYM instance</td>
</tr>
<tr>
<td>Zero</td>
<td>Represents a zero value for an OracleIntervalYM structure</td>
</tr>
</tbody>
</table>

OracleIntervalYM Static Methods
The OracleIntervalYM static methods are listed in Table 9–63.

Table 9–63  OracleIntervalYM Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether two OracleIntervalYM values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether one OracleIntervalYM value is greater than another</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether one OracleIntervalYM value is greater than or equal to another</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether one OracleIntervalYM value is less than another</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether one OracleIntervalYM value is less than or equal to another</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines whether two OracleIntervalYM values are not equal</td>
</tr>
<tr>
<td>Parse</td>
<td>Returns an OracleIntervalYM structure and sets its value for time interval using a string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleIntervalYM with the specified year precision.</td>
</tr>
</tbody>
</table>
OracleIntervalYM Static Operators
The OracleIntervalYM static operators are listed in Table 9–64.

Table 9–64  OracleIntervalYM Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds two OracleIntervalYM values</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines whether two OracleIntervalYM values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines whether one OracleIntervalYM value is greater than another</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines whether one OracleIntervalYM value is greater than or equal to another</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines whether two OracleIntervalYM values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines whether one OracleIntervalYM value is less than another</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines whether one OracleIntervalYM value is less than or equal to another</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts one OracleIntervalYM value from another</td>
</tr>
<tr>
<td>operator -</td>
<td>Negates an OracleIntervalYM structure</td>
</tr>
<tr>
<td>operator *</td>
<td>Multiplies an OracleIntervalYM value by a number</td>
</tr>
<tr>
<td>operator /</td>
<td>Divides an OracleIntervalYM value by a number</td>
</tr>
</tbody>
</table>

OracleIntervalYM Type Conversions
The OracleIntervalYM conversions are listed in Table 9–65.

Table 9–65  OracleIntervalYM Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator long</td>
<td>Converts an OracleIntervalYM structure to a number</td>
</tr>
<tr>
<td>explicit operator OracleIntervalYM</td>
<td>Converts a string to an OracleIntervalYM structure</td>
</tr>
<tr>
<td>implicit operator OracleIntervalYM</td>
<td>Converts the number of months to an OracleIntervalYM structure</td>
</tr>
</tbody>
</table>

OracleIntervalYM Properties
The OracleIntervalYM properties are listed in Table 9–66.

Table 9–66  OracleIntervalYM Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents the Oracle INTERVAL YEAR TO MONTH in an Oracle internal format</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
</tbody>
</table>
OracleIntervalYM Members

Table 9–66 (Cont.) OracleIntervalYM Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Months</td>
<td>Gets the months component of an OracleIntervalYM</td>
</tr>
<tr>
<td>TotalYears</td>
<td>Returns the total number, in years, that represents the period of time in the current OracleIntervalYM structure</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the total number of months that is stored in the OracleIntervalYM structure</td>
</tr>
<tr>
<td>Years</td>
<td>Gets the years component of an OracleIntervalYM</td>
</tr>
</tbody>
</table>

OracleIntervalYM Methods

The OracleIntervalYM methods are listed in Table 9–67.

Table 9–67 OracleIntervalYM Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleIntervalYM instance to the supplied object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object has the same time interval as the current instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleIntervalYM instance</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleIntervalYM structure to a string</td>
</tr>
</tbody>
</table>

See Also:

- “Oracle.DataAccess.Types Namespace” on page 1-4
- 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" on page 1-4
- 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.
OracleIntervalYM Constructors

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// 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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Static Fields

The OracleIntervalYM static fields are listed in Table 9–68.

Table 9–68  OracleIntervalYM Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum value for an OracleIntervalYM structure</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum value for an OracleIntervalYM structure</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an OracleIntervalYM instance</td>
</tr>
<tr>
<td>Zero</td>
<td>Represents a zero value for an OracleIntervalYM structure</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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.
OracleIntervalYM Static Fields

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Static Methods

The OracleIntervalYM static methods are listed in Table 9–69.

### Table 9–69  OracleIntervalYM Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether two OracleIntervalYM values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether one OracleIntervalYM value is greater than another</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether one OracleIntervalYM value is greater than or equal to another</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether one OracleIntervalYM value is less than another</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether one OracleIntervalYM value is less than or equal to another</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines whether two OracleIntervalYM values are not equal</td>
</tr>
<tr>
<td>Parse</td>
<td>Returns an OracleIntervalYM structure and sets its value for time interval using a string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleIntervalYM with the specified year precision.</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members

**Equals**

This static method determines whether two OracleIntervalYM values are equal.

**Declaration**

```csharp
// 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.
OracleIntervalYM Static Methods

- 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members

GreaterThan
This static method determines whether the first of two OracleIntervalYM values is greater than the second.

Declaration

```csharp
// 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 OracleIntervalYMs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// C#
public static bool GreaterThanOrEqual(OracleIntervalYM val1, OracleIntervalYM val2);
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Structure

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" on page 1-4
- `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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members

**NotEquals**

This static method determines whether two `OracleIntervalYM` values are not equal.

**Declaration**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
  ■ OracleIntervalYM Structure
  ■ OracleIntervalYM Members
OracleIntervalYM Static Operators

The OracleIntervalYM static operators are listed in Table 9–70.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds two OracleIntervalYM values</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines whether two OracleIntervalYM values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines whether one OracleIntervalYM value is greater than another</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines whether one OracleIntervalYM value is greater than or equal to another</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines whether two OracleIntervalYM values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines whether one OracleIntervalYM value is less than another</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines whether one OracleIntervalYM value is less than or equal to another</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts one OracleIntervalYM value from another</td>
</tr>
<tr>
<td>operator -</td>
<td>Negates an OracleIntervalYM structure</td>
</tr>
<tr>
<td>operator *</td>
<td>Multiplies an OracleIntervalYM value by a number</td>
</tr>
<tr>
<td>operator /</td>
<td>Divides an OracleIntervalYM value by a number</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator +

This static operator adds two OracleIntervalYM values.

Declaration

```csharp
// 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" on page 1-4
- `OracleIntervalYM` Structure
- `OracleIntervalYM` Members

### operator ==

This static operator determines if two `OracleIntervalYM` values are equal.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleIntervalYM` Structure
- `OracleIntervalYM` Members

### operator >

This static operator determines if the first of two `OracleIntervalYM` values is greater than the second.

**Declaration**

```csharp
// 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" on page 1-4
- 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 OracleIntervalYMs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members
**operator !=**

This static operator determines whether two `OracleIntervalYM` values are not equal.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleIntervalYM Structure`
- `OracleIntervalYM Members`

**operator <**

This static operator determines if the first of two `OracleIntervalYM` values is less than the second.

**Declaration**

```csharp
// 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 OracleIntervalYMs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleIntervalYMs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- `OracleIntervalYM Structure`
- `OracleIntervalYM Members`

**operator -**

This static operator negates an `OracleIntervalYM` structure.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleIntervalYM Structure`
- `OracleIntervalYM Members`

**operator ***

This static operator multiplies an `OracleIntervalYM` value by a number.
Declaration

```csharp
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" on page 1-4
- `OracleIntervalYM Structure`
- `OracleIntervalYM Members`

operator /

This static operator divides an `OracleIntervalYM` value by a number.

Declaration

```csharp
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" on page 1-4
- `OracleIntervalYM Structure`
- `OracleIntervalYM Members`
OracleIntervalYM Type Conversions

The OracleIntervalYM conversions are listed in Table 9–71.

Table 9–71 OracleIntervalYM Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator long</td>
<td>Converts an OracleIntervalYM structure to a number</td>
</tr>
<tr>
<td>explicit operator OracleIntervalYM</td>
<td>Converts a string to an OracleIntervalYM structure</td>
</tr>
<tr>
<td>implicit operator OracleIntervalYM</td>
<td>Converts the number of months to an OracleIntervalYM structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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 OracleIntervalYM 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" on page 1-4
- 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Properties

The OracleIntervalYM properties are listed in Table 9–72.

Table 9–72  OracleIntervalYM Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents the Oracle INTERVAL YEAR TO MONTH in an Oracle internal format</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>Months</td>
<td>Gets the months component of an OracleIntervalYM</td>
</tr>
<tr>
<td>TotalYears</td>
<td>Returns the total number, in years, that represents the period of time in the current OracleIntervalYM structure</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the total number of months that is stored in the OracleIntervalYM structure</td>
</tr>
<tr>
<td>Years</td>
<td>Gets the years component of an OracleIntervalYM</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members

IsNull

This property indicates whether the value has a null value.

Declaration

```csharp
// C#
```
public bool IsNull {get;}

Property Value
Returns true if value has a null value; otherwise, returns false.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members
Value

This property gets the total number of months that is stored in the OracleIntervalYM structure.

Declaration

```csharp
// 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members

Years

This property gets the years component of an OracleIntervalYM.

Declaration

```csharp
// 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" on page 1-4
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Methods

The `OracleIntervalYM` methods are listed in Table 9–73.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CompareTo</code></td>
<td>Compares the current <code>OracleIntervalYM</code> instance to the supplied object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified <code>object</code> has the same time interval as the current instance (Overloaded)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Returns a hash code for the <code>OracleIntervalYM</code> instance</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Inherited from <code>Object</code></td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Converts the current <code>OracleIntervalYM</code> structure to a string</td>
</tr>
</tbody>
</table>

### 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`.  

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleIntervalYM` Structure
- `OracleIntervalYM` Members
Remarks
The following rules apply to the behavior of this method.

■ The comparison must be between OracleIntervalYMs. 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 OracleIntervalYMs that contain a null value are equal.

See Also:
■ "Oracle.DataAccess.Types Namespace" on page 1-4
■ 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 OracleIntervalYMs that contain a null value are equal.

See Also:
■ "Oracle.DataAccess.Types Namespace" on page 1-4
■ 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" on page 1-4
- 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" on page 1-4
- 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#
using System;
using Oracle.DataAccess.Types;

class OracleStringSample
{
    static void Main()
    {
        // Initialize OracleString structs
        OracleString string1 = new OracleString("AAA");

        // Display the string "AAA"
        Console.WriteLine("(0) has length of (1)", string1, string1.Length);

        // Contatenate characters to string1 until the length is 5
        while (string1.Length < 5)
        {
            string1 = OracleString.Concat(string1, "a");

            // Display the string of "AAAaa"
            Console.WriteLine("(0) has length of (1)", string1, string1.Length);
        }
    }
}

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 9–74.

Table 9–74 OracleString Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleString Constructors</td>
<td>Instantiates a new instance of OracleString structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleString Static Fields
The OracleString static fields are listed in Table 9–75.

Table 9–75 OracleString Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleString structure</td>
</tr>
</tbody>
</table>

OracleString Static Methods
The OracleString static methods are listed in Table 9–76.

Table 9–76 OracleString Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concat</td>
<td>Concatenates two OracleString instances and returns a new OracleString instance that represents the result</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines if two OracleString values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether the first of two OracleString values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether the first of two OracleString values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether the first of two OracleString values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether the first of two OracleString values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines whether two OracleString values are not equal</td>
</tr>
</tbody>
</table>

OracleString Static Operators
The OracleString static operators are listed in Table 9–77.
OracleString Members

Table 9–77  **OracleString Static Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Concatenates two OracleString values</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleString values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleString values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleString values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if the two OracleString values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleString values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if two OracleString values are not equal</td>
</tr>
</tbody>
</table>

OracleString Type Conversions
The OracleString type conversions are listed in Table 9–78.

Table 9–78  **OracleString Type Conversions**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator string</td>
<td>Converts the supplied OracleString to a string instance</td>
</tr>
<tr>
<td>implicit operator OracleString</td>
<td>Converts the supplied string to an OracleString instance</td>
</tr>
</tbody>
</table>

OracleString Properties
The OracleString properties are listed in Table 9–79.

Table 9–79  **OracleString Properties**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsCaseIgnored</td>
<td>Indicates whether case should be ignored when performing string comparison</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>Item</td>
<td>Obtains the particular character in an OracleString using an index.</td>
</tr>
<tr>
<td>Length</td>
<td>Returns the length of the OracleString</td>
</tr>
</tbody>
</table>

OracleString Methods
The OracleString methods are listed in Table 9–80.

Table 9–80  **OracleString Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Returns a copy of the current OracleString instance</td>
</tr>
</tbody>
</table>
### OracleString Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleString instance to the supplied object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same string value as the current OracleString structure (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleString instance</td>
</tr>
<tr>
<td>GetNonUnicodeBytes</td>
<td>Returns an array of bytes, containing the contents of the OracleString, in the client character set format</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetUnicodeBytes</td>
<td>Returns an array of bytes, containing the contents of the OracleString, in Unicode format</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleString instance to a string</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleString Structure
- OracleString Members

**OracleString(string)**
This constructor creates a new instance of the OracleString structure and sets its value using a string.

**Declaration**
```csharp
// C#
public OracleString(string data);
```

**Parameters**
- `data`
A string value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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.

**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" on page 1-4
- OracleString Structure
- OracleString Members
OracleString Static Fields

The `OracleString` static fields are listed in Table 9–81.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the <code>OracleString</code> structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleString` Structure
- `OracleString` Members

Null

This static field represents a null value that can be assigned to an instance of the `OracleString` structure.

Declaration

```csharp
public static readonly OracleString Null;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleString` Structure
- `OracleString` Members
OracleString Static Methods

The OracleString static methods are listed in Table 9–82.

Table 9–82  OracleString Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concat</td>
<td>Concatenates two OracleString instances and returns a new OracleString instance that represents the result</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines if two OracleString values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether the first of two OracleString values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether the first of two OracleString values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether the first of two OracleString values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether the first of two OracleString values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines whether two OracleString values are not equal</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleString Structure
- OracleString Members

Concat

This static method concatenates two OracleString instances and returns a new OracleString instance that represents the result.

Declaration

```csharp
// 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.
**Equals**

Overloads Object

This static method determines whether the two `OracleString` values 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 `OracleString` values 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 `OracleString` values that contain a null value are equal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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`.  

OracleString Static Methods

- **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" on page 1-4
- 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**
```csharp
// 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.
### LessThan

This static method determines whether the first of two `OracleString` values is less than the second.

#### Declaration

```csharp
// 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`es that contain a null value are equal.

### LessThanOrEqual

This static method determines whether the first of two `OracleString` values is less than or equal to the second.

#### Declaration

```csharp
// C#
public static bool LessThanOrEqual(OracleString str1, OracleString str2);
```

#### Parameters

- **str1**
  - First `OracleString`.
- **str2**
  - Second `OracleString`.

#### See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleString Structure`
- `OracleString Members`
OracleString Static Methods

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" on page 1-4
- 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" on page 1-4
- OracleString Structure
- OracleString Members
OracleString Static Operators

The OracleString static operators are listed in Table 9–83.

Table 9–83  OracleString Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Concatenates two OracleString values</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleString values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleString values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleString values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if the two OracleString values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleString values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if two OracleString values are not equal</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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 `OracleStrings` that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleStrings that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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**

```csharp
// 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 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" on page 1-4
- OracleString Structure
- OracleString Members
OracleString Type Conversions

The OracleString type conversions are listed in Table 9–84.

Table 9–84  OracleString Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator string</td>
<td>Converts the supplied OracleString to a string instance</td>
</tr>
<tr>
<td>implicit operator OracleString</td>
<td>Converts the supplied string to an OracleString instance</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- `OracleString` Structure
- `OracleString` Members
OracleString Properties

The `OracleString` properties are listed in Table 9–85.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsCaseIgnored</td>
<td>Indicates whether case should be ignored when performing string comparison</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>Item</td>
<td>Obtains the particular character in an <code>OracleString</code> using an index.</td>
</tr>
<tr>
<td>Length</td>
<td>Returns the length of the <code>OracleString</code></td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleString Structure`
- `OracleString Members`

### IsCaseIgnored

This property indicates whether case should be ignored when performing string comparison.

**Declaration**

```csharp
// C#
public bool IsCaseIgnored {get;set;}
```

**Property Value**

Returns `true` if string comparison must ignore case; otherwise `false`.

**Remarks**

Default value is `true`.

**Example**

```csharp
// C#
using System;
using Oracle.DataAccess.Types;

class IsCaseIgnoredSample
{
    static void Main()
    {
        OracleString string1 = new OracleString("aAaAa");
        OracleString string2 = new OracleString("AaAaA");

        // Ignore case for comparisons
        string1.IsCaseIgnored = true;
        string2.IsCaseIgnored = true;

        // Same; Prints 0
        Console.WriteLine(string1.CompareTo(string2));
    }
}
```
// Make comparisons case sensitive
// Note that IsCaseIgnored must be set to false for both
// OracleStrings; otherwise an exception is thrown
string1.IsCaseIgnored = false;
string2.IsCaseIgnored = false;

// Different; Prints non-zero value
Console.WriteLine(string1.CompareTo(string2));
}
}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 a null value; otherwise, returns false.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
OracleString Properties

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleString Structure
- OracleString Members
OracleString Methods

The OracleString methods are listed in Table 9–86.

**Table 9–86  OracleString Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Returns a copy of the current OracleString instance</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleString instance to the supplied object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same string value as the current OracleString structure (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleString instance</td>
</tr>
<tr>
<td>GetNonUnicodeBytes</td>
<td>Returns an array of bytes, containing the contents of the OracleString, in the client character set format</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetUnicodeBytes</td>
<td>Returns an array of bytes, containing the contents of the OracleString, in Unicode format</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleString instance to a string</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleString Structure
- OracleString Members

**Clone**

This method creates a copy of an OracleString instance.

**Declaration**

```csharp
// 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**

```csharp
// C#

using System;
using Oracle.DataAccess.Types;

class CloneSample
{
    static void Main()
    {
        OracleString str1 = new OracleString("aAaAa");
    }
}
```
OracleString str2 = str1.Clone();

// The OracleStrings are same; Prints 0
Console.WriteLine(str1.CompareTo(str2));
}
}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// 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 OracleStrings. 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 OracleStrings that contain a null value are equal.
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 OracleStrings that contain a null value are equal.

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.
GetNonUnicodeBytes

This method returns an array of bytes, containing the contents of the OracleString, in the client character set format.

**Declaration**

```csharp
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" on page 1-4
- OracleString Structure
- OracleString Members

GetUnicodeBytes

This method returns an array of bytes, containing the contents of the OracleString in Unicode format.

**Declaration**

```csharp
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" on page 1-4
- 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" on page 1-4
- 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

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#
using System;
using Oracle.DataAccess.Types;

class OracleTimeStampSample
{
    static void Main()
    {
        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" on page 1-4
- 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 9–87

Table 9–87 OracleTimeStamp Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleTimeStamp Constructors</td>
<td>Instantiates a new instance of OracleTimeStamp structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTimeStamp Static Fields
The OracleTimeStamp static fields are listed in Table 9–88.

Table 9–88 OracleTimeStamp Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an OracleTimeStamp structure, which is December 31, 9999 23:59:59.999999999</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an OracleTimeStamp structure, which is January 1, -4712 0:0:0</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleTimeStamp structure</td>
</tr>
</tbody>
</table>

OracleTimeStamp Static Methods
The OracleTimeStamp static methods are listed in Table 9–89.

Table 9–89 OracleTimeStamp Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleTimeStamp values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleTimeStamp values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleTimeStamp values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleTimeStamp values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleTimeStamp values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleTimeStamp values are not equal</td>
</tr>
<tr>
<td>GetSysDate</td>
<td>Gets an OracleTimeStamp structure that represents the current date and time</td>
</tr>
<tr>
<td>Parse</td>
<td>Gets an OracleTimeStamp structure and sets its value using the supplied string</td>
</tr>
</tbody>
</table>
**OracleTimeStamp Static Operators**
The OracleTimeStamp static operators are listed in Table 9–90.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds the supplied instance value to the supplied OracleTimeStamp and returns a new OracleTimeStamp structure (Overloaded)</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleTimeStamp values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleTimeStamp values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleTimeStamp values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if the two OracleTimeStamp values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleTimeStamp values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleTimeStamp values is less than or equal to the second</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts the supplied instance value from the supplied OracleTimeStamp and returns a new OracleTimeStamp structure (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleTimeStamp Static Type Conversions**
The OracleTimeStamp static type conversions are listed in Table 9–91.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator OracleTimeStamp</td>
<td>Converts an instance value to an OracleTimeStamp structure (Overloaded)</td>
</tr>
<tr>
<td>implicit operator OracleTimeStamp</td>
<td>Converts an instance value to an OracleTimeStamp structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator DateTime</td>
<td>Converts an OracleTimeStamp value to a DateTime structure</td>
</tr>
</tbody>
</table>

**OracleTimeStamp Properties**
The OracleTimeStamp properties are listed in Table 9–92.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents an Oracle TIMESTAMP in Oracle internal format</td>
</tr>
</tbody>
</table>
OracleTimeStamp Members

Table 9–93  (Cont.) OracleTimeStamp Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day</td>
<td>Specifies the day component of an OracleTimeStamp</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the OracleTimeStamp instance has a null value</td>
</tr>
<tr>
<td>Hour</td>
<td>Specifies the hour component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Millisecond</td>
<td>Specifies the millisecond component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Minute</td>
<td>Specifies the minute component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Month</td>
<td>Specifies the month component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Nanosecond</td>
<td>Specifies the nanosecond component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Second</td>
<td>Specifies the second component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the date and time that is stored in the OracleTimeStamp structure</td>
</tr>
<tr>
<td>Year</td>
<td>Specifies the year component of an OracleTimeStamp</td>
</tr>
</tbody>
</table>

OracleTimeStamp Methods
The OracleTimeStamp methods are listed in Table 9–93.

Table 9–93  OracleTimeStamp Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDays</td>
<td>Adds the supplied number of days to the current instance</td>
</tr>
<tr>
<td>AddHours</td>
<td>Adds the supplied number of hours to the current instance</td>
</tr>
<tr>
<td>AddMilliseconds</td>
<td>Adds the supplied number of milliseonds to the current instance</td>
</tr>
<tr>
<td>AddMinutes</td>
<td>Adds the supplied number of minutes to the current instance</td>
</tr>
<tr>
<td>AddMonths</td>
<td>Adds the supplied number of months to the current instance</td>
</tr>
<tr>
<td>AddNanoseconds</td>
<td>Adds the supplied number of nanoseconds to the current instance</td>
</tr>
<tr>
<td>AddSeconds</td>
<td>Adds the supplied number of seconds to the current instance</td>
</tr>
<tr>
<td>AddYears</td>
<td>Adds the supplied number of years to the current instance</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleTimeStamp instance to an object, and returns an integer that represents their relative values</td>
</tr>
</tbody>
</table>
Table 9–93 (Cont.) OracleTimeStamp Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same date and time as the current OracleTimeStamp instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleTimeStamp instance</td>
</tr>
<tr>
<td>GetDaysBetween</td>
<td>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</td>
</tr>
<tr>
<td>GetYearsBetween</td>
<td>Subtracts value1 from the current instance and returns an OracleIntervalYM that represents the difference between value1 and the current instance using OracleIntervalYM</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToOracleDate</td>
<td>Converts the current OracleTimeStamp structure to an OracleDate structure</td>
</tr>
<tr>
<td>ToOracleTimeStampLTZ</td>
<td>Converts the current OracleTimeStamp structure to an OracleTimeStampLTZ structure</td>
</tr>
<tr>
<td>ToOracleTimeStampTZ</td>
<td>Converts the current OracleTimeStamp structure to an OracleTimeStampTZ structure</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleTimeStamp structure to a string</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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)
  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, 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)
  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" on page 1-4
- 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);
OracleTimeStamp Structure

Parameters

- \( dt \)
  
The supplied DateTime value.

Exceptions

ArgumentException - The \( dt \) parameter cannot be used to construct a valid OracleTimeStamp.

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#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleTimeStampSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStamp(string) constructor
    
```
OracleGlobalization info = OracleGlobalization.GetClientInfo();
info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(info);

// 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
info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
OracleGlobalization.SetThreadInfo(info);

// Prints "1999-NOV-11 11:02:33.444000000 AM"
Console.WriteLine(ts.ToString());

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
- 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).
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.

Declarations

// 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" on page 1-4
- 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.
OracleTimeStamp Constructors

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" on page 1-4
- 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, second, and nanosecond.

Declaration

// C#
public OracleTimeStamp (int year, int month, int day, int hour,  
    int minute, int second, int nanosecond);
**OracleTimeStamp Structure**

### 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).

### 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**

```csharp
// 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" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Static Fields

The OracleTimeStamp static fields are listed in Table 9–94.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an OracleTimeStamp structure, which is December 31, 9999 23:59:59.999999999</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an OracleTimeStamp structure, which is January 1, -4712 0:0:0</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleTimeStamp structure</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// C#
public static readonly OraTimestamp MaxValue;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// C#
public static readonly OracleTimeStamp MinValue;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Static Methods

The OracleTimeStamp static methods are listed in Table 9–95.

Table 9–95 OracleTimeStamp Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleTimeStamp values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleTimeStamp values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleTimeStamp values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleTimeStamp values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleTimeStamp values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleTimeStamp values are not equal</td>
</tr>
<tr>
<td>GetSysDate</td>
<td>Gets an OracleTimeStamp structure that represents the current date and time</td>
</tr>
<tr>
<td>Parse</td>
<td>Gets an OracleTimeStamp structure and sets its value using the supplied string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleTimeStamp with the specified fractional second precision</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
OracleTimeStamp Static Methods

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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
■ 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.
OracleTimeStamp Static Methods

- 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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members

**NotEquals**

This static method determines if two OracleTimeStamp values are not equal.

**Declaration**

```csharp
// 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 `OracleTimeStamp`s that contain a null value are equal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`

**GetSysDate**

This static method gets an `OracleTimeStamp` structure that represents the current date and time.

**Declaration**

```csharp
// C#
public static OracleTimeStamp GetSysDate();
```

**Return Value**

An `OracleTimeStamp` structure that represents the current date and time.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`

**Parse**

This static method gets an `OracleTimeStamp` structure and sets its value using the supplied string.

**Declaration**

```csharp
// 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**

```csharp
// C#
using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ParseSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the Parse() method
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // 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
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
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" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Static Operators

The OracleTimeStamp static operators are listed in Table 9–96.

Table 9–96 OracleTimeStamp Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds the supplied instance value to the supplied OracleTimeStamp and returns a new OracleTimeStamp structure (Overloaded)</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleTimeStamp values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleTimeStamp values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleTimeStamp values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if the two OracleTimeStamp values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleTimeStamp values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleTimeStamp values is less than or equal to the second</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts the supplied instance value from the supplied OracleTimeStamp and returns a new OracleTimeStamp structure (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace” on page 1-4
- 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.
operator + (OracleTimeStamp, OracleIntervalDS)

This static operator adds the supplied OracleIntervalDS to the OracleTimeStamp and returns a new OracleTimeStamp structure.

Declaration

```csharp
// 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" on page 1-4
- 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

```csharp
// C#
public static operator + (OracleTimeStamp value1, OracleIntervalYM value2);
```

Parameters

- `value1`
  - An OracleTimeStamp.
- `value2`
  - An OracleIntervalYM.

Return Value

An OracleTimeStamp.
OracleTimeStamp Static Operators

Remarks
If either parameter has a null value, the returned OracleTimeStamp has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
■ 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" on page 1-4
■ 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" on page 1-4
- 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" on page 1-4
- `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" on page 1-4
- `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

```csharp
// 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 `OracleTimeStamp`s that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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 returns 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.
operator - (OracleTimeStamp, OracleIntervalDS)

This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStamp value, and returns a new OracleTimeStamp structure.

Declaration

```csharp
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" on page 1-4
- 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

```csharp
public static operator -(OracleTimeStamp value1, OracleIntervalYM value2);
```

Parameters

- `value1` An OracleTimeStamp.
- `value2` An OracleIntervalYM instance.

Return Value

An OracleTimeStamp structure.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
Remarks
If either parameter has a null value, the returned OracleTimeStamp has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeSpan Structure
- OracleTimeSpan 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" on page 1-4
- OracleTimeSpan Structure
- OracleTimeSpan Members
OracleTimeStamp Static Type Conversions

The OracleTimeStamp static type conversions are listed in Table 9–97.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator OracleTimeStamp</td>
<td>Converts an instance value to an OracleTimeStamp structure (Overloaded)</td>
</tr>
<tr>
<td>implicit operator OracleTimeStamp</td>
<td>Converts an instance value to an OracleTimeStamp structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator DateTime</td>
<td>Converts an OracleTimeStamp value to a DateTime structure</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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);
OracleTimeStamp Static Type Conversions

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" on page 1-4
- 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" on page 1-4
- 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**

```csharp
// C#
using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleTimeStampSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the explicit
        // operator OracleTimeStamp(string)
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // 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
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints '1999-NOV-11 11:02:33.444000000 AM'
        Console.WriteLine(ts.ToString());
    }
}
OracleTimeStamp Static Type Conversions

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
- 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" on page 1-4
- 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.
implicit operator OracleTimeStamp(DateTime)

This static type conversion operator converts a DateTime value to an OracleTimeStamp structure.

Declaration

// C#
public static implicit operator OracleTimeSpan(DateTime value);

Parameters

■ value

A DateTime instance.

Return Value

An OracleTimeStamp structure.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-4
■ 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(OracleTimeSpan value);

Parameters

■ value

An OracleTimeSpan instance.

Return Value

A DateTime containing the date and time in the current instance.

Exceptions

OracleNullValueException - The OracleTimeSpan structure has a null value.

Remarks

The precision of the OracleTimeStamp can be lost during the conversion.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Properties

The OracleTimeStamp properties are listed in Table 9–98.

### Table 9–98 OracleTimeStamp Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents an Oracle TIMESTAMP in Oracle internal format</td>
</tr>
<tr>
<td>Day</td>
<td>Specifies the day component of an OracleTimeStamp</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the OracleTimeStamp instance has a null value</td>
</tr>
<tr>
<td>Hour</td>
<td>Specifies the hour component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Millisecond</td>
<td>Specifies the millisecond component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Minute</td>
<td>Specifies the minute component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Month</td>
<td>Specifies the month component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Nanosecond</td>
<td>Specifies the nanosecond component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Second</td>
<td>Specifies the second component of an OracleTimeStamp</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the date and time that is stored in the OracleTimeStamp structure</td>
</tr>
<tr>
<td>Year</td>
<td>Specifies the year component of an OracleTimeStamp</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members

**BinData**

This property returns an array of bytes that represents an Oracle TIMESTAMP in Oracle internal format.

**Declaration**

```csharp
// C#
publci 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" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Properties

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.

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" on page 1-4
- 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.
OracleTimeStamp Structure

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" on page 1-4
- 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" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members

Month

This property gets the month component of an OracleTimeStamp.

Declaration

// C#
public int Month{get;}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
**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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
**Value**

This property specifies the date and time that is stored in the `OracleTimeStamp` structure.

**Declaration**

```csharp
// C#
public DateTime Value { get; }
```

**Property Value**

A `DateTime`.

**Exceptions**

`OracleNullValueException` - The current instance has a null value.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`

**Year**

This property gets the year component of an `OracleTimeStamp`.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`
OracleTimeStamp Methods

The OracleTimeStamp methods are listed in Table 9–99.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDays</td>
<td>Adds the supplied number of days to the current instance</td>
</tr>
<tr>
<td>AddHours</td>
<td>Adds the supplied number of hours to the current instance</td>
</tr>
<tr>
<td>AddMilliseconds</td>
<td>Adds the supplied number of milliseconds to the current instance</td>
</tr>
<tr>
<td>AddMinutes</td>
<td>Adds the supplied number of minutes to the current instance</td>
</tr>
<tr>
<td>AddMonths</td>
<td>Adds the supplied number of months to the current instance</td>
</tr>
<tr>
<td>AddNanoseconds</td>
<td>Adds the supplied number of nanoseconds to the current instance</td>
</tr>
<tr>
<td>AddSeconds</td>
<td>Adds the supplied number of seconds to the current instance</td>
</tr>
<tr>
<td>AddYears</td>
<td>Adds the supplied number of years to the current instance</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleTimeStamp instance to an object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same date and time as the current OracleTimeStamp instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleTimeStamp instance</td>
</tr>
<tr>
<td>GetDaysBetween</td>
<td>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</td>
</tr>
<tr>
<td>GetYearsBetween</td>
<td>Subtracts value1 from the current instance and returns an OracleIntervalYM that represents the difference between value1 and the current instance using OracleIntervalYM</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToOracleDate</td>
<td>Converts the current OracleTimeStamp structure to an OracleDate structure</td>
</tr>
<tr>
<td>ToOracleTimeStampLTZ</td>
<td>Converts the current OracleTimeStamp structure to an OracleTimeStampLTZ structure</td>
</tr>
<tr>
<td>ToOracleTimeStampTZ</td>
<td>Converts the current OracleTimeStamp structure to an OracleTimeStampTZ structure</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleTimeStamp structure to a string</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 \((-1,000,000,000 < \text{days} < 1,000,000,000)\)

Return Value

An OracleTimeStamp.

Exceptions

- ArgumentException - The argument value is out of the specified range.
- OracleNullValueException - The current instance has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 \((-24,000,000,000 < \text{hours} < 24,000,000,000)\).

Return Value

An OracleTimeStamp.

Exceptions

- ArgumentException - The argument value is out of the specified range.
- OracleNullValueException - The current instance has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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.
AddMonths

This method adds the supplied number of months to the current instance.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`
OracleTimeStamp Methods

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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.
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" on page 1-4
- 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" on page 1-4
■ 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" on page 1-4
■ 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(OracleStamp 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" on page 1-4
■ 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.
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" on page 1-4
■ 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" on page 1-4
■ 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” on page 1-4
■ OracleTimeStamp Structure
■ OracleTimeStamp Members
■ "OracleGlobalization Class" on page 6-2
■ "Globalization Support” on page 3-53

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#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ToStringSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStamp(string)
        // constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        infoTimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // 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
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
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#
using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleTimeStampLTZSample
{
    static void Main()
    {
        // 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);
    }
}
OracleTimeStampLTZ Structure

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 9–100

Table 9–100 OracleTimeStampLTZ Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleTimeStampLTZ Constructors</td>
<td>Instantiates a new instance of OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTimeStampLTZ Static Fields
The OracleTimeStampLTZ static fields are listed in Table 9–101.

Table 9–101 OracleTimeStampLTZ Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an OracleTimeStampLTZ structure, which is December 31, 9999 23:59:59.999999999</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an OracleTimeStampLTZ structure, which is January 1, -4712 0:0:0</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleTimeStampLTZ structure</td>
</tr>
</tbody>
</table>

OracleTimeStampLTZ Static Methods
The OracleTimeStampLTZ static methods are listed in Table 9–102.

Table 9–102 OracleTimeStampLTZ Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleTimeStampLTZ values are equal (Overloaded)</td>
</tr>
<tr>
<td>GetLocalTimeZoneName</td>
<td>Gets the client's local time zone name</td>
</tr>
<tr>
<td>GetLocalTimeZoneOffset</td>
<td>Gets the client's local time zone offset relative to UTC</td>
</tr>
<tr>
<td>GetSysDate</td>
<td>Gets an OracleTimeStampLTZ structure that represents the current date and time</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleTimeStampLTZ values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleTimeStampLTZ values is less than the second</td>
</tr>
</tbody>
</table>
OracleTimeStampLTZ Members

OracleTimeStampLTZ Members

Table 9–102  (Cont.) OracleTimeStampLTZ Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleTimeStampLTZ values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleTimeStampLTZ values are not equal</td>
</tr>
<tr>
<td>Parse</td>
<td>Gets an OracleTimeStampLTZ structure and sets its value for date and time using the supplied string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleTimeStampLTZ with the specified fractional second precision</td>
</tr>
</tbody>
</table>

OracleTimeStampLTZ Static Type Operators

The OracleTimeStampLTZ static type operators are listed in Table 9–103.

Table 9–103  OracleTimeStampLTZ Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator+</td>
<td>Adds the supplied instance value to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleTimeStampLTZ values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleTimeStampLTZ values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if two OracleTimeStampLTZ values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleTimeStampLTZ values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleTimeStampLTZ values is less than or equal to the second</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts the supplied instance value from the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTimeStampLTZ Static Type Conversions

The OracleTimeStampLTZ static type conversions are listed in Table 9–104.

Table 9–104  OracleTimeStampLTZ Static Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator OracleTimeStampLTZ</td>
<td>Converts an instance value to an OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
</tbody>
</table>
OracleTimeStampLTZ Structure

Table 9–104 (Cont.) OracleTimeStampLTZ Static Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>implicit operator OracleTimeStampLTZ</td>
<td>Converts an instance value to an OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator DateTime</td>
<td>Converts an OracleTimeStampLTZ value to a DateTime structure</td>
</tr>
</tbody>
</table>

OracleTimeStampLTZ Properties

The OracleTimeStampLTZ properties are listed in Table 9–105.

Table 9–105 OracleTimeStampLTZ Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE in Oracle internal format</td>
</tr>
<tr>
<td>Day</td>
<td>Specifies the day component of an OracleTimeStampLTZ</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the OracleTimeStampLTZ instance has a null value</td>
</tr>
<tr>
<td>Hour</td>
<td>Specifies the hour component of an OracleTimeStampLTZ</td>
</tr>
<tr>
<td>Millisecond</td>
<td>Specifies the millisecond component of an OracleTimeStampLTZ</td>
</tr>
<tr>
<td>Minute</td>
<td>Specifies the minute component of an OracleTimeStampLTZ</td>
</tr>
<tr>
<td>Month</td>
<td>Specifies the month component of an OracleTimeStampLTZ</td>
</tr>
<tr>
<td>Nanosecond</td>
<td>Specifies the nanosecond component of an OracleTimeStampLTZ</td>
</tr>
<tr>
<td>Second</td>
<td>Specifies the second component of an OracleTimeStampLTZ</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the date and time that is stored in the OracleTimeStampLTZ structure</td>
</tr>
<tr>
<td>Year</td>
<td>Specifies the year component of an OracleTimeStampLTZ</td>
</tr>
</tbody>
</table>

OracleTimeStampLTZ Methods

The OracleTimeStampLTZ methods are listed in Table 9–106.

Table 9–106 OracleTimeStampLTZ Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDays</td>
<td>Adds the supplied number of days to the current instance</td>
</tr>
<tr>
<td>AddHours</td>
<td>Adds the supplied number of hours to the current instance</td>
</tr>
<tr>
<td>AddMilliseconds</td>
<td>Adds the supplied number of milliseconds to the current instance</td>
</tr>
</tbody>
</table>
### Table 9–106 (Cont.) OracleTimeStampLTZ Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddMinutes</td>
<td>Adds the supplied number of minutes to the current instance</td>
</tr>
<tr>
<td>AddMonths</td>
<td>Adds the supplied number of months to the current instance</td>
</tr>
<tr>
<td>AddNanoseconds</td>
<td>Adds the supplied number of nanoseconds to the current instance</td>
</tr>
<tr>
<td>AddSeconds</td>
<td>Adds the supplied number of seconds to the current instance</td>
</tr>
<tr>
<td>AddYears</td>
<td>Adds the supplied number of years to the current instance</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleTimeStampLTZ instance to an object and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same date and time as the current OracleTimeStampLTZ instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleTimeStampLTZ instance</td>
</tr>
<tr>
<td>GetDaysBetween</td>
<td>Subtracts an OracleTimeStampLTZ from the current instance and returns an OracleIntervalDS that represents the difference</td>
</tr>
<tr>
<td>GetYearsBetween</td>
<td>Subtracts an OracleTimeStampLTZ from the current instance and returns an OracleIntervalYM that represents the difference</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToOracleDate</td>
<td>Converts the current OracleTimeStampLTZ structure to an OracleDate structure</td>
</tr>
<tr>
<td>ToOracleTimeStamp</td>
<td>Converts the current OracleTimeStampLTZ structure to an OracleTimeStamp structure</td>
</tr>
<tr>
<td>ToOracleTimeStampTZ</td>
<td>Converts the current OracleTimeStampLTZ structure to an OracleTimeStampTZ structure</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleTimeStampLTZ structure to a string</td>
</tr>
<tr>
<td>ToUniversalTime</td>
<td>Converts the current local time to Coordinated Universal Time (UTC)</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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)**
  
  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" on page 1-4
- 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

```csharp
// C#
public OracleTimeStampLTZ (DateTime dt);
```
OracleTimeStampLTZ Constructors

**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" on page 1-4
- 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#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleTimeStampLTZSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStampLTZ(string) // constructor
    
}
OracleGlobalization info = OracleGlobalization.GetClientInfo();
    info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
    OracleGlobalization.SetThreadInfo(info);

    // 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
    info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
    OracleGlobalization.SetThreadInfo(info);

    // Prints '1999-NOV-11 11:02:33.444000000 AM'
    Console.WriteLine(ts.ToString());
}
OracleTimeStampLTZ Constructors

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

- 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" on page 1-4
- 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 999,999,999).

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 OracleTimeStampLTZ (that is, the day is out of range for the month).

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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, second, and nanosecond.

Declaration

// C#
public OracleTimeStampLTZ (int year, int month, int day, int hour,
    int minute, int second, int nanosecond);
OracleTimeStampLTZ Constructors

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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Static Fields

The OracleTimeStampLTZ static fields are listed in Table 9–107.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an OracleTimeStampLTZ structure, which is December 31, 9999 23:59:59.999999999</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an OracleTimeStampLTZ structure, which is January 1, -4712 0:0:0</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleTimeStampLTZ structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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.
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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Static Methods

The OracleTimeStampLTZ static methods are listed in Table 9–108.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleTimeStampLTZ values are equal (Overloaded)</td>
</tr>
<tr>
<td>GetLocalTimeZoneName</td>
<td>Gets the client's local time zone name</td>
</tr>
<tr>
<td>GetLocalTimeZoneOffset</td>
<td>Gets the client's local time zone offset relative to UTC</td>
</tr>
<tr>
<td>GetSysDate</td>
<td>Gets an OracleTimeStampLTZ structure that represents the current date and time</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleTimeStampLTZ values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleTimeStampLTZ values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleTimeStampLTZ values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleTimeStampLTZ values are not equal</td>
</tr>
<tr>
<td>Parse</td>
<td>Gets an OracleTimeStampLTZ structure and sets its value for date and time using the supplied string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleTimeStampLTZ with the specified fractional second precision</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Equals

This static method determines if two OracleTimeStampLTZ values are equal.

Declaration

```csharp
// 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 OracleTimeStampLTZs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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 GetLocalTimeZoneOffset();

Return Value
A TimeSpan structure containing the local time zone hours and time zone minutes.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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 OracleTimeStampLTZs that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleTimeStampLTZs that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- `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

```csharp
// 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" on page 1-4
- `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" on page 1-4
■ 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.

**Example**

```csharp
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class Parse
{
    static void Main()
    {
        // Set the nls_timestamp_format for the Parse() method
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // construct OracleTimeStampLTZ from a string using the format specified.
        OracleTimeStampLTZ ts =
            OracleTimeStampLTZ.Parse("11-NOV-1999 11:02:33.444 AM");

        // Set the nls_timestamp_format for the ToString() method
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}
```

**See Also:**
- "Oracle.DataAccess.Types Namespace” on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class” on page 6-2
- "Globalization Support” on page 3-53

**SetPrecision**

This static method returns a new instance of an `OracleTimeStampLTZ` with the specified fractional second precision.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleTimeStampLTZ Structure`
- `OracleTimeStampLTZ Members`
OracleTimeStampLTZ Static Type Operators

The OracleTimeStampLTZ static type operators are listed in Table 9–109.

Table 9–109 OracleTimeStampLTZ Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator+</td>
<td>Adds the supplied instance value to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleTimeStampLTZ values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleTimeStampLTZ values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleTimeStampLTZ values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if two OracleTimeStampLTZ values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleTimeStampLTZ values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleTimeStampLTZ values is less than or equal to the second</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts the supplied instance value from the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
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" on page 1-4
- 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.
OracleTimeStampLTZ Static Type Operators

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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- `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.
operator >=

This static operator determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleTimeStampLTZ` Structure
- `OracleTimeStampLTZ` Members

operator !=

This static operator determines if two `OracleTimeStampLTZ` values are not equal.

**Declaration**

```csharp
// C#
public static bool operator != (OracleTimeStampLTZ value1,
                                OracleTimeStampLTZ value2);
```

**Parameters**

- `value1`  
  First `OracleTimeStampLTZ`.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleTimeStampLTZ` Structure
- `OracleTimeStampLTZ` Members
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 OracleTimeStampLTZs that contain a null value are equal.

**operator <**
This static operator determines if the first of two OracleTimeStampLTZ values is less than the second.

**Declaration**
```csharp
// 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 OracleTimeStampLTZs that contain a null value are equal.
operator <=

This static operator determines if the first of two OracleTimeStampLTZ values is less than or equal to the second.

**Declaration**

```csharp
// 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 OracleTimeStampLTZs that contain a null value are equal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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, OracleIntervalDS)**

This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStampLTZ value, and returns a new OracleTimeStampLTZ structure.

**Declaration**

```csharp
// 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" on page 1-4
- 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**

```csharp
// C#
```

```csharp
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
```
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" on page 1-4
■ 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" on page 1-4
■ OracleTimeStampLTZ Structure
■ OracleTimeStampLTZ Members
OracleTimeStampLTZ Static Type Conversions

The OracleTimeStampLTZ static type conversions are listed in Table 9–110.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator OracleTimeStampLTZ</td>
<td>Converts an instance value to an OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
<tr>
<td>implicit operator OracleTimeStampLTZ</td>
<td>Converts an instance value to an OracleTimeStampLTZ structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator DateTime</td>
<td>Converts an OracleTimeStampLTZ value to a DateTime structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ(OracleTimeStamp)

This static type conversion operator converts an OracleTimeStamp value to an OracleTimeStampLTZ structure.

Declaration

```csharp
// C#
public static explicit operator OracleTimeStampLTZ (OracleTimeStamp value1);
```
OracleTimeStampLTZ Static Type Conversions

**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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

**explicit operator OracleTimeStampLTZ(OracleTimeStampTZ)**

This static type conversion operator converts an OracleTimeStampTZ value to an OracleTimeStampLTZ structure.

**Declaration**

```csharp
// 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" on page 1-4
- 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#
using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class OracleTimeStampLTZSample
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStampLTZ(string) constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // 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
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints '1999-NOV-11 11:02:33.444000000 AM'
        Console.WriteLine(ts.ToString());
    }
}
implicit operator OracleTimeStampLTZ

This static type conversion operator 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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
- Oracle Database SQL Reference for further information on datetime format elements
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

**implicit operator OracleTimeStampLTZ(DateTime)**

This static type conversion operator converts a DateTime structure to an OracleTimeStampLTZ structure.

**Declaration**

```csharp
// C#
public static implicit operator OracleTimeStampLTZ(DateTime value1);
```

**Parameters**

- `value1`  
  A DateTime structure.

**Return Value**

An OracleTimeStampLTZ structure.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

**explicit operator DateTime**

This static type conversion operator converts an OracleTimeStampLTZ value to a DateTime structure.

**Declaration**

```csharp
// 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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Properties

The OracleTimeStampLTZ properties are listed in Table 9–111.

### Table 9–111 OracleTimeStampLTZ Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE in Oracle internal format.</td>
</tr>
<tr>
<td>Day</td>
<td>Specifies the day component of an OracleTimeStampLTZ.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the OracleTimeStampLTZ instance has a null value.</td>
</tr>
<tr>
<td>Hour</td>
<td>Specifies the hour component of an OracleTimeStampLTZ.</td>
</tr>
<tr>
<td>Millisecond</td>
<td>Specifies the millisecond component of an OracleTimeStampLTZ.</td>
</tr>
<tr>
<td>Minute</td>
<td>Specifies the minute component of an OracleTimeStampLTZ.</td>
</tr>
<tr>
<td>Month</td>
<td>Specifies the month component of an OracleTimeStampLTZ.</td>
</tr>
<tr>
<td>Nanosecond</td>
<td>Specifies the nanosecond component of an OracleTimeStampLTZ.</td>
</tr>
<tr>
<td>Second</td>
<td>Specifies the second component of an OracleTimeStampLTZ.</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the date and time that is stored in the OracleTimeStampLTZ structure.</td>
</tr>
<tr>
<td>Year</td>
<td>Specifies the year component of an OracleTimeStampLTZ.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Properties

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.

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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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 represents a minute. Range of Minute is (0 to 59).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Month

This property gets the month component of an OracleTimeStampLTZ.

Declaration

// C#
public int Month{get;}
OracleTimeStampLTZ Properties

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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
Value

This property specifies the date and time that is stored in the OracleTimeStampLTZ structure.

**Declaration**

```csharp
// C#
public DateTime Value{get;}
```

**Property Value**

A DateTime.

**Exceptions**

OracleNullValueException - The current instance has a null value.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Year

This property gets the year component of an OracleTimeStampLTZ.

**Declaration**

```csharp
// 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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Methods

The OracleTimeStampLTZ methods are listed in Table 9–112.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDays</td>
<td>Adds the supplied number of days to the current instance</td>
</tr>
<tr>
<td>AddHours</td>
<td>Adds the supplied number of hours to the current instance</td>
</tr>
<tr>
<td>AddMilliseconds</td>
<td>Adds the supplied number of milliseconds to the current instance</td>
</tr>
<tr>
<td>AddMinutes</td>
<td>Adds the supplied number of minutes to the current instance</td>
</tr>
<tr>
<td>AddMonths</td>
<td>Adds the supplied number of months to the current instance</td>
</tr>
<tr>
<td>AddNanoseconds</td>
<td>Adds the supplied number of nanoseconds to the current instance</td>
</tr>
<tr>
<td>AddSeconds</td>
<td>Adds the supplied number of seconds to the current instance</td>
</tr>
<tr>
<td>AddYears</td>
<td>Adds the supplied number of years to the current instance</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleTimeStampLTZ instance to an object and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same date and time as the current OracleTimeStampLTZ instance (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleTimeStampLTZ instance</td>
</tr>
<tr>
<td>GetDaysBetween</td>
<td>Subtracts an OracleTimeStampLTZ from the current instance and returns an OracleIntervalDS that represents the difference</td>
</tr>
<tr>
<td>GetYearsBetween</td>
<td>Subtracts an OracleTimeStampLTZ from the current instance and returns an OracleIntervalYM that represents the difference</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToOracleDate</td>
<td>Converts the current OracleTimeStampLTZ structure to an OracleDate structure</td>
</tr>
<tr>
<td>ToOracleTimeStamp</td>
<td>Converts the current OracleTimeStampLTZ structure to an OracleTimeStamp structure</td>
</tr>
<tr>
<td>ToOracleTimeStampTZ</td>
<td>Converts the current OracleTimeStampLTZ structure to an OracleTimeStampTZ structure</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleTimeStampLTZ structure to a string</td>
</tr>
<tr>
<td>ToUniversalTime</td>
<td>Converts the current local time to Coordinated Universal Time (UTC)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
ArgumentOutOfRangeAssertionException - The argument value is out of the specified range.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-4
■ 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.
ArgumentOutOfRangeAssertionException - The argument value is out of the specified range.
AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

**Declaration**

```csharp
// C#
public OracleTimeStampLTZ AddMilliseconds(double milliseconds);
```

**Parameters**

- `milliseconds`

  The supplied number of milliseconds. Range is \((-8.64 * 10^{16}<milliseconds<8.64 * 10^{16})\).

**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" on page 1-4
- `OracleTimeStampLTZ Structure`
- `OracleTimeStampLTZ Members`

AddMinutes

This method adds the supplied number of minutes to the current instance.

**Declaration**

```csharp
// 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.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleTimeStampLTZ Structure`
- `OracleTimeStampLTZ Members`
ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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.
OracleTimeStampLTZ Methods

Exceptions
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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 OracleTimeStampLTZs. 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 OracleTimeStampLTZs that contain a null value are equal.
OracleTimeStampLTZ Methods

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// 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 `OracleTimeStampLTZs` that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

### GetHashCode

**Overrides** Object

This method returns a hash code for the OracleTimeStampLTZ instance.

**Declaration**

```csharp
// C#
public override int GetHashCode();
```

**Return Value**

A number that represents the hash code.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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.
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" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

ToString

Overrrides 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**

```csharp
// C#

using System;
using Oracle.DataAccess.Types;
using Oracle.DataAccess.Client;

class ToString
{
    static void Main()
    {
        // Set the nls_timestamp_format for the OracleTimeStampLTZ(string)
        // constructor
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // 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
        info.TimeStampFormat = "YYYY-MON-DD HH:MI:SS.FF AM";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM"
        Console.WriteLine(ts.ToString());
    }
}
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

**ToUniversalTime**

This method converts the current local time to Coordinated Universal Time (UTC).

**Declaration**

```csharp
// 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" on page 1-4
- 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.

Example

// C#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleTimeStampTZSample
{
    static void Main()
    {
        // Set the nls parameters for the current thread
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeZone = "US/Eastern";
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        info.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // 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" on page 1-4
- 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 9–113.

**Table 9–113  OracleTimeStampTZ Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleTimeStampTZ Constructors</td>
<td>Instantiates a new instance of OracleTimeStampTZ structure (Overloaded)</td>
</tr>
</tbody>
</table>

**OracleTimeStampTZ Static Fields**

The OracleTimeStampTZ static fields are listed in Table 9–114.

**Table 9–114  OracleTimeStampTZ Static Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an OracleTimeStampTZ structure in UTC, which is December 31, 999923:59:59.999999999</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an OracleTimeStampTZ structure in UTC, which is January 1, -4712 0:0:0</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleTimeStampTZ structure</td>
</tr>
</tbody>
</table>

**OracleTimeStampTZ Static Methods**

The OracleTimeStampTZ static methods are listed in Table 9–115.

**Table 9–115  OracleTimeStampTZ Static Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleTimeStampTZ values are equal (Overloaded)</td>
</tr>
<tr>
<td>GetSysDate</td>
<td>Gets an OracleTimeStampTZ structure that represents the current date and time</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleTimeStampTZ values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleTimeStampTZ values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleTimeStampTZ values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleTimeStampTZ values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleTimeStampTZ values are not equal</td>
</tr>
</tbody>
</table>
OracleTimeStampTZ Structure

Table 9–115 (Cont.) OracleTimeStampTZ Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>Gets an OracleTimeStampTZ structure and sets its value for date and time using the supplied string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleTimeStampTZ with the specified fractional second precision</td>
</tr>
</tbody>
</table>

OracleTimeStampTZ Static Operators
The OracleTimeStampTZ static operators are listed in Table 9–116.

Table 9–116 OracleTimeStampTZ Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds the supplied instance value to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure (Overloaded)</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleTimeStampTZ values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleTimeStampTZ values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleTimeStampTZ values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if two OracleTimeStampTZ values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleTimeStampTZ values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleTimeStampTZ values is less than or equal to the second</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts the supplied instance value from the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTimeStampTZ Static Type Conversions
The OracleTimeStampTZ static type conversions are listed in Table 9–117.

Table 9–117 OracleTimeStampTZ Static Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator OracleTimeStampTZ</td>
<td>Converts an instance value to an OracleTimeStampTZ structure (Overloaded)</td>
</tr>
<tr>
<td>implicit operator OracleTimeStampTZ</td>
<td>Converts an instance value to an OracleTimeStampTZ structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator DateTime</td>
<td>Converts an OracleTimeStampTZ value to a DateTime structure in the current time zone</td>
</tr>
</tbody>
</table>
OracleTimeStampTZ Properties

The OracleTimeStampTZ properties are listed in Table 9–118.

Table 9–118  OracleTimeStampTZ Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format</td>
</tr>
<tr>
<td>Day</td>
<td>Specifies the day component of an OracleTimeStampTZ in the current time zone</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value</td>
</tr>
<tr>
<td>Hour</td>
<td>Specifies the hour component of an OracleTimeStampTZ in the current time zone</td>
</tr>
<tr>
<td>Millisecond</td>
<td>Specifies the millisecond component of an OracleTimeStampTZ in the current time zone</td>
</tr>
<tr>
<td>Minute</td>
<td>Specifies the minute component of an OracleTimeStampTZ in the current time zone</td>
</tr>
<tr>
<td>Month</td>
<td>Specifies the month component of an OracleTimeStampTZ in the current time zone</td>
</tr>
<tr>
<td>Nanosecond</td>
<td>Specifies the nanosecond component of an OracleTimeStampTZ in the current time zone</td>
</tr>
<tr>
<td>Second</td>
<td>Specifies the second component of an OracleTimeStampTZ in the current time zone</td>
</tr>
<tr>
<td>TimeZone</td>
<td>Returns the time zone of the OracleTimeStampTZ instance</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the date and time that is stored in the OracleTimeStampTZ structure in the current time zone</td>
</tr>
<tr>
<td>Year</td>
<td>Specifies the year component of an OracleTimeStampTZ</td>
</tr>
</tbody>
</table>

OracleTimeStampTZ Methods

The OracleTimeStampTZ methods are listed in Table 9–119.

Table 9–119  OracleTimeStampTZ Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDays</td>
<td>Adds the supplied number of days to the current instance</td>
</tr>
<tr>
<td>AddHours</td>
<td>Adds the supplied number of hours to the current instance</td>
</tr>
<tr>
<td>AddMilliseconds</td>
<td>Adds the supplied number of milliseconds to the current instance</td>
</tr>
<tr>
<td>AddMinutes</td>
<td>Adds the supplied number of minutes to the current instance</td>
</tr>
<tr>
<td>AddMonths</td>
<td>Adds the supplied number of months to the current instance</td>
</tr>
<tr>
<td>AddNanoseconds</td>
<td>Adds the supplied number of nanoseconds to the current instance</td>
</tr>
</tbody>
</table>
### Table 9–119 (Cont.) **OracleTimeStampTZ Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSeconds</td>
<td>Adds the supplied number of seconds to the current instance</td>
</tr>
<tr>
<td>AddYears</td>
<td>Adds the supplied number of years to the current instance</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleTimeStampTZ instance to an object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same date and time as the current OracleTimeStampTZ instance</td>
</tr>
<tr>
<td>GetDaysBetween</td>
<td>Subtracts an OracleTimeStampTZ from the current instance and returns an OracleIntervalDS that represents the time interval</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleTimeStampTZ instance</td>
</tr>
<tr>
<td>GetTimeZoneOffset</td>
<td>Gets the time zone information in hours and minutes of the current OracleTimeStampTZ</td>
</tr>
<tr>
<td>GetYearsBetween</td>
<td>Subtracts an OracleTimeStampTZ from the current instance and returns an OracleIntervalYM that represents the time interval</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToLocalTime</td>
<td>Converts the current OracleTimeStampTZ instance to local time</td>
</tr>
<tr>
<td>ToOracleDate</td>
<td>Converts the current OracleTimeStampTZ structure to an OracleDate structure</td>
</tr>
<tr>
<td>ToOracleTimeStampLTZ</td>
<td>Converts the current OracleTimeStampTZ structure to an OracleTimeStampLTZ structure</td>
</tr>
<tr>
<td>ToOracleTimeStamp</td>
<td>Converts the current OracleTimeStampTZ structure to an OracleTimeStamp structure</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleTimeStampTZ structure to a string</td>
</tr>
<tr>
<td>ToUniversalTime</td>
<td>Converts the current datetime to Coordinated Universal Time (UTC)</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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)
  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, 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" on page 1-4
  - `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**
```csharp
// 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" on page 1-4
  - `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**
```csharp
// C#
public OracleTimeStampTZ (DateTime value1, string timeZone);
```

**Parameters**
- `value1`
  The supplied `DateTime` value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`
OracleTimeStampTZ Constructors

- **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 timeZone 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" on page 1-4
- 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#
using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleTimeStampTZSample
{
    static void Main()
    {
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // 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
        info.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // Prints "1999-NOV-11 11:02:33.444000000 AM US/Pacific"
        Console.WriteLine(tstz.ToString());
    }
}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
- 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).
OracleTimeStampTZ Constructors

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" on page 1-4
- 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 timeZone is null, the OracleGlobalization.TimeZone of the thread is used.
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.

Note: PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by OracleTimeStampTZ.
OracleTimeStampTZ Constructors

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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.
OracleTimeStampTZ Structure

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.

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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeStampTZ(int, int, int, int, int, int, double)
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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**

```csharp
// 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.
OracleTimeStampTZ Constructors

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 $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.

**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**

```csharp
// 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" on page 1-4
- 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**

```csharp
// 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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
OracleTimeStampTZ Static Fields

The OracleTimeStampTZ static fields are listed in Table 9–120.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an OracleTimeStampTZ structure in UTC, which is December 31, 999923:59:59.999999999</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an OracleTimeStampTZ structure in UTC, which is January 1, -4712 0:0:0</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an instance of the OracleTimeStampTZ structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// C#
public static readonly OracleTimeStampTZ MaxValue;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// C#
public static readonly OracleTimeStampTZ MinValue;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
Null

This static field represents a null value that can be assigned to an instance of the `OracleTimeStampTZ` structure.

**Declaration**

```csharp
// C#
public static readonly OracleTimeStampTZ Null;
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`
OracleTimeStampTZ Static Methods

The OracleTimeStampTZ static methods are listed in Table 9–121.

Table 9–121    OracleTimeStampTZ Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines if two OracleTimeStampTZ values are equal (Overloaded)</td>
</tr>
<tr>
<td>GetSysDate</td>
<td>Gets an OracleTimeStampTZ structure that represents the current date and time</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines if the first of two OracleTimeStampTZ values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines if the first of two OracleTimeStampTZ values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines if the first of two OracleTimeStampTZ values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines if the first of two OracleTimeStampTZ values is less than or equal to the second</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines if two OracleTimeStampTZ values are not equal</td>
</tr>
<tr>
<td>Parse</td>
<td>Gets an OracleTimeStampTZ structure and sets its value for date and time using the supplied string</td>
</tr>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleTimeStampTZ with the specified fractional second precision</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Equals

This static method determines if two OracleTimeStampTZ values are equal.

Declaration

```csharp
// 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" on page 1-4
- `OracleTimeStampTZ` Structure
- `OracleTimeStampTZ` Members

GetSysDate
This static method gets an `OracleTimeStampTZ` structure that represents the current date and time.

Declaration
```csharp
// C#
public static OracleTimeStampTZ GetSysDate();
```

Return Value
An `OracleTimeStampTZ` structure that represents the current date and time.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleTimeStampTZ` Structure
- `OracleTimeStampTZ` Members

GreaterThan
This static method determines if the first of two `OracleTimeStampTZ` values is greater than the second.

Declaration
```csharp
// 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 `OracleTimeStampTZs` that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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 `OracleTimeStampTZs` that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `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.

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" on page 1-4
- 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.
OracleTimeStampTZ Static Methods

- 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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

NotEquals

This static method determines if two OracleTimeStampTZ values are not equal.

Declaration

```csharp
// 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 OracleTimeStampTZs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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

```csharp
// 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

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

class ParseSample
{
    static void Main()
    {
        // Set the nls_timestamp_tz_format for the Parse() method
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // 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
        info.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // Prints '1999-NOV-11 11:02:33.444000000 AM US/Pacific'
        Console.WriteLine(tstz.ToString());
    }
}
```
**OracleTimeStampTZ Static Methods**

---

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

**SetPrecision**

This static method returns a new instance of an OracleTimeStampTZ with the specified fractional second precision.

**Declaration**

```csharp
// 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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
OracleTimeStampTZ Static Operators

The OracleTimeStampTZ static operators are listed in Table 9–122.

Table 9–122 OracleTimeStampTZ Static Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds the supplied instance value to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure (Overloaded)</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if two OracleTimeStampTZ values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleTimeStampTZ values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleTimeStampTZ values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if two OracleTimeStampTZ values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleTimeStampTZ values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleTimeStampTZ values is less than or equal to the second</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts the supplied instance value from the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- 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

```csharp
// 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" on page 1-4
- `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" on page 1-4
- `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`
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 OracleTimeStampTZs that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleTimeStampTZs that contain a null value are equal.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-4
■ 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.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-4
■ OracleTimeStampTZ Structure
■ OracleTimeStampTZ Members
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 OracleTimeStampTZs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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 OracleTimeStampTZs that contain a null value are equal.
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 OracleTimeStampTZs that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator - (OracleTimeStampTZ, OracleIntervalDS)

This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampTZ` value, 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` structure.

Remarks

If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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,

OracleTimeStampTZ Structure

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" on page 1-4
■ 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" on page 1-4
■ OracleTimeStampTZ Structure
■ OracleTimeStampTZ Members
OracleTimeStampTZ Static Type Conversions

The **OracleTimeStampTZ** static type conversions are listed in Table 9–123.

### Table 9–123  OracleTimeStampTZ Static Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator OracleTimeStampTZ</td>
<td>Converts an instance value to an OracleTimeStampTZ structure (Overloaded)</td>
</tr>
<tr>
<td>implicit operator OracleTimeStampTZ</td>
<td>Converts an instance value to an OracleTimeStampTZ structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator DateTime</td>
<td>Converts an OracleTimeStampTZ value to a DateTime structure in the current time zone</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

### 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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

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.
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.TimeUnitTZFormat 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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class OracleTimeStampTZSample
{
    static void Main()
    {
        // Set the nls_timestamp_tz_format for the explicit operator
        // OracleTimeStampTZ(string)
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeUnitTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);
OracleTimeStampTZ Structure

// 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
info.TimeStampTZFormat = "YYYY-MON-DD HH:MI:SS.FF AM TZR";
OracleGlobalization.SetThreadInfo(info);
Console.WriteLine(tstz.ToString());
}
}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

**implicit operator OracleTimeStampTZ**

**Declaration**

// C#
public static implicit operator OracleTimeStampTZ(OracleDate value1);

**Parameters**

- **value1**

**implicit operator OracleTimeStampTZ(OracleDate)**

This static type conversion operator converts an OracleDate value 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" on page 1-4
  - OracleTimeStampTZ Structure
  - OracleTimeStampTZ Members
  - "OracleGlobalization Class" on page 6-2
  - "Globalization Support" on page 3-53

**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 OracleTimeZone structure also has a null value.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeZone Structure
- OracleTimeZone Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

**implicit operator OracleTimeZoneTZ(DateTime)**
This static type conversion operator converts a DateTime structure to an OracleTimeZoneTZ structure.

**Declaration**
// C#
public static implicit operator OracleTimeZoneTZ (DateTime value1);

**Parameters**
- value1
  A DateTime structure.

**Return Value**
The returned OracleTimeZoneTZ 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 TimeZoneTZ structure.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeZoneTZ Structure
- OracleTimeZoneTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53
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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
OracleTimeStampTZ Properties

The OracleTimeStampTZ properties are listed in Table 9–124.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinData</td>
<td>Returns an array of bytes that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format.</td>
</tr>
<tr>
<td>Day</td>
<td>Specifies the day component of an OracleTimeStampTZ in the current time zone.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether the current instance has a null value.</td>
</tr>
<tr>
<td>Hour</td>
<td>Specifies the hour component of an OracleTimeStampTZ in the current time zone.</td>
</tr>
<tr>
<td>Millisecond</td>
<td>Specifies the millisecond component of an OracleTimeStampTZ in the current time zone.</td>
</tr>
<tr>
<td>Minute</td>
<td>Specifies the minute component of an OracleTimeStampTZ in the current time zone.</td>
</tr>
<tr>
<td>Month</td>
<td>Specifies the month component of an OracleTimeStampTZ in the current time zone.</td>
</tr>
<tr>
<td>Nanosecond</td>
<td>Specifies the nanosecond component of an OracleTimeStampTZ in the current time zone.</td>
</tr>
<tr>
<td>Second</td>
<td>Specifies the second component of an OracleTimeStampTZ in the current time zone.</td>
</tr>
<tr>
<td>TimeZone</td>
<td>Returns the time zone of the OracleTimeStampTZ instance.</td>
</tr>
<tr>
<td>Value</td>
<td>Returns the date and time that is stored in the OracleTimeStampTZ structure in the current time zone.</td>
</tr>
<tr>
<td>Year</td>
<td>Specifies the year component of an OracleTimeStampTZ.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// 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.
Day

This property specifies the day component of an `OracleTimeStampTZ` in the current time zone.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`

IsNull

This property indicates whether the current instance has a null value.

**Declaration**

```csharp
// 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" on page 1-4
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`

Hour

This property specifies the hour component of an `OracleTimeStampTZ` in the current time zone.

**Declaration**

```csharp
// 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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

**Millissecond**
This property gets the millisecond component of an OracleTimeStampTZ in the current time zone.

**Declaration**
```csharp
// C#
public double Millisecond{get;}
```

**Property Value**
A number that represents a millisecond. Range of **Milliseconds** is (0 to 999.999999)

**Exceptions**
OracleNullValueException - The current instance has a null value.

*See Also:*
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

**Minute**
This property gets the minute component of an OracleTimeStampTZ in the current time zone.

**Declaration**
```csharp
// C#
public int Minute{get;}
```

**Property Value**
A number that represents a minute. Range of **Minute** is (0 to 59).

**Exceptions**
OracleNullValueException - The current instance has a null value.

*See Also:*
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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).
OracleTimeStampTZ Properties

**Exceptions**

OracleNullValueException - The current instance has a null value.

**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" on page 1-4
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`

**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" on page 1-4
- `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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
OracleTimeStampTZ Methods

The OracleTimeStampTZ methods are listed in Table 9–125.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDays</td>
<td>Adds the supplied number of days to the current instance</td>
</tr>
<tr>
<td>AddHours</td>
<td>Adds the supplied number of hours to the current instance</td>
</tr>
<tr>
<td>AddMilliseconds</td>
<td>Adds the supplied number of milliseconds to the current instance</td>
</tr>
<tr>
<td>AddMinutes</td>
<td>Adds the supplied number of minutes to the current instance</td>
</tr>
<tr>
<td>AddMonths</td>
<td>Adds the supplied number of months to the current instance</td>
</tr>
<tr>
<td>AddNanoseconds</td>
<td>Adds the supplied number of nanoseconds to the current instance</td>
</tr>
<tr>
<td>AddSeconds</td>
<td>Adds the supplied number of seconds to the current instance</td>
</tr>
<tr>
<td>AddYears</td>
<td>Adds the supplied number of years to the current instance</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Compares the current OracleTimeStampTZ instance to an object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether an object has the same date and time as the current OracleTimeStampTZ instance (Overloaded)</td>
</tr>
<tr>
<td>GetDaysBetween</td>
<td>Subtracts an OracleTimeStampTZ from the current instance and returns an OracleIntervalDS that represents the time interval</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the OracleTimeStamp instance</td>
</tr>
<tr>
<td>GetTimeZoneOffset</td>
<td>Gets the time zone information in hours and minutes of the current OracleTimeStampTZ</td>
</tr>
<tr>
<td>GetYearsBetween</td>
<td>Subtracts an OracleTimeStampTZ from the current instance and returns an OracleIntervalYM that represents the time interval</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToLocalTime</td>
<td>Converts the current OracleTimeStampTZ instance to local time</td>
</tr>
<tr>
<td>ToOracleDate</td>
<td>Converts the current OracleTimeStampTZ structure to an OracleDate structure</td>
</tr>
<tr>
<td>ToOracleTimeStampLTZ</td>
<td>Converts the current OracleTimeStampTZ structure to an OracleTimeStampLTZ structure</td>
</tr>
<tr>
<td>ToOracleTimeStamp</td>
<td>Converts the current OracleTimeStampTZ structure to an OracleTimeStamp structure</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current OracleTimeStampTZ structure to a string</td>
</tr>
<tr>
<td>ToUniversalTime</td>
<td>Converts the current datetime to Coordinated Universal Time (UTC)</td>
</tr>
</tbody>
</table>
**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

### AddDays

This method adds the supplied number of days to the current instance.

**Declaration**

```csharp
// C#
public OracleTimeStampTZ AddDays(double days);
```

**Parameters**

- **days**
  
  The supplied number of days. Range is \((-1,000,000,000 < \text{days} < 1,000,000,000)\)

**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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

### AddHours

This method adds the supplied number of hours to the current instance.

**Declaration**

```csharp
// C#
public OracleTimeStampTZ AddHours(double hours);
```

**Parameters**

- **hours**
  
  The supplied number of hours. Range is \((-24,000,000,000 < \text{hours} < 24,000,000,000)\).

**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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```csharp
// C#
public OracleTimeStampTZ AddMilliseconds(double milliseconds);
```

Parameters

- `milliseconds`
  
  The supplied number of milliseconds. Range is (-8.64 * 10^16 < milliseconds < 8.64 * 10^16).

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" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```csharp
// 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
■ 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 * 10^13 < seconds < 8.64 * 10^13).

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" on page 1-4
■ 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" on page 1-4
- `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**

```csharp
// 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 `OracleTimeStampTZs`. 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.
OracleTimeStampTZ Methods

- Two OracleTimeStampTZs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
■ 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" on page 1-4
■ 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" on page 1-4
- `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#

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

class ToStringSample
{
    static void Main()
    {
        // Set the nls parameters for the current thread
        OracleGlobalization info = OracleGlobalization.GetClientInfo();
        info.TimeZone = "US/Eastern";
        info.TimeStampFormat = "DD-MON-YYYY HH:MI:SS.FF AM";
        info.TimeStampTZFormat = "DD-MON-YYYY HH:MI:SS.FF AM TZR";
        OracleGlobalization.SetThreadInfo(info);

        // Create an OracleTimeStampTZ in US/Pacific time zone
OracleTimeStampTZ t1 = 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 t2 = ts.ToOracleTimeStampTZ();

// Calculate the difference between t1 and t2
OracleIntervalDS idsDiff = t1.GetDaysBetween(t2);

// Prints 'US/Pacific'
Console.WriteLine("t1.TimeZone = " + t1.TimeZone);

// Prints 'US/Eastern'
Console.WriteLine("t2.TimeZone = " + t2.TimeZone);

// Prints 3
Console.WriteLine("idsDiff.Hours = " + idsDiff.Hours);

// Prints 0
Console.WriteLine("idsDiff.Minutes = " + idsDiff.Minutes);

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 6-2
- "Globalization Support" on page 3-53

**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:

- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
This section covers the Oracle Data Provider for .NET Types exceptions.

This chapter contains these topics:

- 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" on page 1-4
- 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 10–1.

Table 10–1 OracleTypeException Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleTypeException Constructors</td>
<td>Creates a new instance of the OracleTypeException class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTypeException Static Methods
The OracleTypeException static methods are listed in Table 10–2.

Table 10–2 OracleTypeException Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTypeException Properties
The OracleTypeException properties are listed in Table 10–3.

Table 10–3 OracleTypeException Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the error messages that occur in the exception</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the name of the data provider that generates the error</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from Exception</td>
</tr>
</tbody>
</table>

OracleTypeException Methods
The OracleTypeException methods are listed in Table 10–4.

Table 10–4 OracleTypeException Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns the fully qualified name of this exception</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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, StreamingContext)
  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" on page 1-4
- 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" on page 1-4
- OracleTypeException Class
- OracleTypeException Members

OracleTypeException(SerializationInfo, StreamingContext)

This constructor creates a new instance of the OracleTypeException class with the specified serialization information, si, and the specified streaming context, sc.

Declaration

// C#
protected OracleException {SerializationInfo si, StreamingContext sc};

Parameters

- si
  The specified serialization information.
OracleTypeException Constructors

- $sc$
  The specified streaming context.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTypeException Class
- OracleTypeException Members
OracleTypeException Static Methods

The OracleTypeException static methods are listed in Table 10–5.

Table 10–5  OracleTypeException Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTypeException Class
- OracleTypeException Members
OracleTypeException Properties

The OracleTypeException properties are listed in Table 10–6.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the error messages that occur in the exception</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the name of the data provider that generates the error</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from Exception</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTypeException Class
- OracleTypeException Members

Message

Overrides Exception

This property specifies the error messages that occur in the exception.

Declaration

```csharp
// C#
public override string Message {get;}
```

Property Value

An error message.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTypeException Class
- OracleTypeException Members

Source

Overrides Exception

This property specifies the name of the data provider that generates the error.

Declaration

```csharp
// C#
public override string Source {get;}
```

Property Value

Oracle Data Provider for .NET.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTypeException Class
- OracleTypeException Members
OracleTypeException Methods

The OracleTypeException methods are listed in Table 10–7.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns the fully qualified name of this exception</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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**

```csharp
// C#
public override string ToString();
```

**Return Value**

The fully qualified name of this exception.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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 10–8.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleNullValueException Constructor</td>
<td>Creates a new instance of the OracleNullValueException class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleNullValueException Static Methods
The OracleNullValueException static methods are listed in Table 10–9.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleNullValueException Properties
The OracleNullValueException properties are listed in Table 10–10.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>Message</td>
<td>Inherited from OracleTypeException</td>
</tr>
<tr>
<td>Source</td>
<td>Inherited from OracleTypeException</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from Exception</td>
</tr>
</tbody>
</table>

OracleNullValueException Methods
The OracleNullValueException methods are listed in Table 10–11.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from OracleTypeException</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
  - 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" on page 1-4
- 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);
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleNullValueException Class
- OracleNullValueException Members
OracleNullValueException Static Methods

The OracleNullValueException static methods are listed in Table 10–12.

Table 10–12  OracleNullValueException Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleNullValueException Class
- OracleNullValueException Members
OracleNullValueException Properties

The `OracleNullValueException` properties are listed in Table 10–13.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from <code>Exception</code></td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from <code>Exception</code></td>
</tr>
<tr>
<td>Message</td>
<td>Inherited from <code>OracleTypeException</code></td>
</tr>
<tr>
<td>Source</td>
<td>Inherited from <code>OracleTypeException</code></td>
</tr>
<tr>
<td>StackTrace</td>
<td>Inherited from <code>Exception</code></td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from <code>Exception</code></td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- `OracleNullValueException` Class
- `OracleNullValueException` Members
OracleNullValueException Methods

The OracleNullValueException methods are listed in Table 10–14.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from OracleTypeException</td>
</tr>
</tbody>
</table>
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" on page 1-4
- 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 10–15.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleTruncateException Constructors</td>
<td>Creates a new instance of the OracleTruncateException class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTruncateException Static Methods
The OracleTruncateException static methods are listed in Table 10–16.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleTruncateException Properties
The OracleTruncateException properties are listed in Table 10–17.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>Message</td>
<td>Inherited from OracleExceptionException</td>
</tr>
<tr>
<td>Source</td>
<td>Inherited from OracleExceptionException</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from Exception</td>
</tr>
</tbody>
</table>

OracleTruncateException Methods
The OracleTruncateException methods are listed in Table 10–18.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from OracleExceptionException</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- 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" on page 1-4
- OracleTruncateException Class
- OracleTruncateException Members
OracleTruncateException Static Methods

The OracleTruncateException static methods are listed in Table 10–19.

Table 10–19  OracleTruncateException Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTruncateException Class
- OracleTruncateException Members
OracleTruncateException Properties

The OracleTruncateException properties are listed in Table 10–20.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>Message</td>
<td>Inherited from OracleTypeException</td>
</tr>
<tr>
<td>Source</td>
<td>Inherited from OracleTypeException</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from Exception</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTruncateException Class
- OracleTruncateException Members
OracleTruncateException Methods

The OracleTruncateException methods are listed in Table 10–21.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from OracleTypeException</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-4
- OracleTruncateException Class
- OracleTruncateException Members
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 BLOBS, CLOBs, and NCLOBs while external LOBs include BFILES.

**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
array bind operations, 3-18
  ArrayBindCount, 4-12
  ArrayBindIndex, 4-166
  ArrayBindSize, 4-210
  ArrayBindStatus, 4-212
ArrayBindCount property, 4-12
ArrayBindIndex property, 4-166
ArrayBindSize property, 3-15, 3-19, 4-210
ArrayBindStatus property, 3-15, 3-19, 4-212
assembly, 1-1

B
behavior of ExecuteScalar method for REF CURSOR, 3-32
BFILE, 3-32
BINARY_DOUBLE, 3-11
BINARY_FLOAT, 3-11
binding, 3-10
  PL/SQL Associative Array, 3-15
BLOB, 3-32

C
callback support, 3-8
case-sensitivity
  column name mapping, 3-46
client globalization settings, 3-54, 3-56
CLOB, 3-32
CollectionType property, 3-15
column data
  special characters in, 3-45
  commits
    changes to XML data, 3-48
  connection dependency, 3-37
Connection Lifetime attribute, 3-3
  connection pooling, 3-3
  example, 3-3
Connection property, 3-34
  connection string attributes, 4-63
Connection Timeout attribute, 3-3
ConnectionString attributes, 3-3
  Connection Lifetime, 3-1, 3-4
  Connection Timeout, 3-1, 3-4
Data Source, 3-1
DBA Privilege, 3-1
Decr Pool Size, 3-1, 3-4
Enlist, 3-1
Incr Pool Size, 3-1, 3-4
Max Pool Size, 3-1, 3-4
Min Pool Size, 3-1, 3-4
Password, 3-1
Persist Security Info, 3-1
Pooling, 3-1, 3-4
Proxy Password, 3-1
Proxy User, 3-1
Statement Cache Purge, 3-1
Statement Cache Size, 3-1
User Id, 3-1
Validate Connection, 3-1, 3-4
ConnectionString property, 3-3, 3-4
Constraints property, 3-51
  configuring, 3-52
  controlling query reexecution, 3-51

data loss, 3-49
data manipulation
  using XML, 3-44
database
  changes to, 3-44
DataSet, 3-34
  updating, 3-32
    updating to database, 3-51
DataTable, 3-52
Datatable properties, 3-51
DbType
  inference, 3-12
debug tracing, 3-57
  registry settings, 3-57
Decr Pool Size attribute
  Validate Connection attribute, 3-3
default mapping
  improving, 3-48
documentation, 2-2
Dynamic Help, Visual Studio .Net
documentation, 2-2
error handling
  array bind operations
  error handling, 3-19
example
  connection pooling, 3-3
ExecuteNonQuery method, 3-32
ExecuteScalar method, 3-32

FailoverEvent Enumeration, 7-10
FailoverReturnCode Enumeration, 7-11
FailoverType Enumeration, 7-12
features, 3-1
  new, xxiii
FetchSize property
  fine-tuning, 3-29
  setting at design time, 3-30
  setting at runtime, 3-30
  using, 3-29

Global Assembly Cache (GAC), 2-2
globalization settings, 3-53
  client, 3-54
  session, 3-54
  thread-based, 3-55
globalization support, 3-53
globalization-sensitive operations, 3-56
grid-computing, xxiii, xxiv
  grids, xxiii, xxiv

handling date and time format
  manipulating data in XML, 3-44
  retrieving queries in XML, 3-39

improving default mapping, 3-48
Incr Pool Size attribute, 3-3
inference from Value property, 3-14
inference of DbType and OracleDbType from
  Value, 3-14
inference of DbType from OracleDbType, 3-12
inference of OracleDbType from DbType, 3-13
inference of types, 3-12
InitialLOBFetchSize property, 3-27
InitialLONGFetchSize, 3-27
InitialLONGFetchSize property, 3-27
input binding
  XMLType column, 3-38
installation, 2-2
  ODP.NET, 2-2
  Oracle Data Provider for .NET, 2-2
Integrated Help, 2-2
  interference in OracleParameter class, 3-12

introduction, overview, 1-1
InvalidCastException, 3-23

large binary datatypes, 3-33
large character datatypes, 3-33
LOB Connection property, 3-33
LOBs
temporary, 3-35
updatering, 3-34
LOBs updating, 3-34
LONG and LONG RAW datatypes, 3-33

Max Pool Size attribute, 3-3
metadata, 3-53
  Microsoft Transaction Server, 2-1
Min Pool Size attribute, 3-3
MTS, 2-1
Multiple Oracle Homes, xxiv
  multiple tables
  changes to, 3-48

native XML support, 3-35
NCLOB, 3-32
.NET Framework datatype, 3-22
.NET Stream class, 3-33
.NET type accessors, 3-23
.NET Types
  inference, 3-12
  NULL values
  retrieving from column, 3-44
  number of rows fetched in round-trip
  controlling, 3-29

object-relational data, 3-43
  saving changes from XML data, 3-48
  obtaining a REF CURSOR, 3-31
  obtaining an OracleRefCursor, 3-31
  obtaining data from an OracleDataReader, 3-23
  Obtaining LOB data
  InitialLOBFetchSize property, 3-27
  obtaining LONG and LONG RAW Data, 3-27
OCI
  statement caching, Oracle Call Interface
  statement caching, 3-21

ODP.NET
  installing, 2-2
  ODP.NET LOB classes, 3-32
  ODP.NET Type accessors, 3-26
  ODP.NET Type classes, 3-22
  ODP.NET Type exceptions, 10-1
  ODP.NET Type structures, 3-22, 9-1
  ODP.NET Types, 3-22
  overview, 3-22
ODP.NET XML Support, 3-35
operating system authentication, 3-5
Oracle 8.1.7, 3-35
  saving changes to, 3-46
Oracle Data Provider for .NET
  installing, 2-2
  system requirements, 2-1
Oracle native types, 3-22
  supported by ODP.NET, 3-23
Oracle Services for Microsoft Transaction Server, 2-1
Oracle Universal Installer (OUI), 2-2
Oracle XDK, 3-35
ORACLE_BASE\ORACLE_HOME\bin
directory, 2-2
Oracle9i
  saving changes to, 3-47
Oracle9i XML Developer’s Kits, 3-35
OracleBFile
  constructors, 8-7
  instance methods, 8-17
  instance properties, 8-11
  members, 8-4
  static fields, 8-9
  static methods, 8-10
OracleBFile Class, 8-2
OracleBinary
  constructor, 9-7
  instance methods, 9-26
  members, 9-4
  properties, 9-23
  static fields, 9-8
  static methods, 9-9
  static operators, 9-15
  static type conversion operators, 9-21
OracleBinary Structure, 9-2
OracleBlob
  constructors, 8-40
  instance methods, 8-49
  instance properties, 8-44
  members, 8-37
  static fields, 8-42
  static methods, 8-43
OracleBlob Class, 8-35
OracleClob
  constructors, 8-74
  instance methods, 8-84
  instance properties, 8-78
  members, 8-71
  static fields, 8-76
  static methods, 8-77
OracleClob Class, 8-69
OracleCollectionType Enumeration, 4-281
OracleCommand
  ArrayBindCount, 4-12
  constructors, 4-7
  members, 4-4
  properties, 4-10
  public methods, 4-24
  static methods, 4-9
OracleCommand ArrayBindCount property, 3-18
OracleCommand Class, 4-2
OracleCommand class
  FetchSize property, 3-29
OracleCommand object, 3-10
OracleCommand Transaction object, 3-10
OracleCommandBuilder, 3-53
  constructors, 4-42
  event delegates, 4-53
  events, 4-52
  members, 4-40
  properties, 4-47
  public methods, 4-49
  static methods, 4-44
  updating dataset, 3-51
OracleCommandBuilder Class, 4-38
OracleConnection
  constructors, 4-59
  event delegates, 4-85
  events, 4-83
  members, 4-56
  obtaining a reference, 3-37
  properties, 4-62
  public methods, 4-71
  static methods, 4-61
OracleConnection Class, 4-54
Oracle.DataAccess.dll assembly, 2-2
OracleDataAdapter, 3-49
  constructors, 4-90
  event delegates, 4-108
  events, 4-104
  FillSchema, 3-52, 3-53
  members, 4-88
  properties, 4-94
  public methods, 4-99
  Requery property, 3-51
  SafeMapping Property, 3-50
  SelectCommand, 3-52, 3-53
  static methods, 4-93
OracleDataAdapter Class, 4-86
OracleDataAdapter Safe Type Mapping, 3-49
OracleDataReader, 3-23
  constructors, 4-112
  properties, 4-116
  public methods, 4-123
  static methods, 4-115
OracleDataReader Class, 4-109
OracleDataReader SchemaTable, 4-153
OracleDate
  constructors, 9-34
  members, 9-31
  methods, 9-60
  properties, 9-56
  static fields, 9-39
  static methods, 9-41
  static operators, 9-47
  static type conversions, 9-52
OracleDate Structure, 9-29
OracleDbType
OracleRowUpdatingEventArgs Class, 4-258
OracleRowUpdatingEventHandler Delegate, 4-265
OracleString
constructors, 9-214
members, 9-211
methods, 9-235
properties, 9-232
static fields, 9-219
static methods, 9-220
static operators, 9-225
type conversions, 9-230
OracleString Structure, 9-209
OracleTimeStamp
constructors, 9-246
members, 9-242
methods, 9-282
properties, 9-277
static methods, 9-255
static operators, 9-262
static type conversions, 9-271
OracleTimeStamp Structure, 9-240
OracleTimeStampLTZ
constructors, 9-299
members, 9-295
methods, 9-336
properties, 9-331
static fields, 9-306
static methods, 9-308
static operators, 9-316
static type conversions, 9-325
static type operators, 9-316
OracleTimeStampLTZ Structure, 9-293
OracleTimeStampTZ
constructors, 9-354
members, 9-350
methods, 9-396
properties, 9-390
static fields, 9-366
static methods, 9-368
static operators, 9-375
static type conversions, 9-384
OracleTimeStampTZ Structure, 9-348
OracleTransaction
members, 4-269
properties, 4-271
public methods, 4-273
static methods, 4-270
OracleTruncateException
constructors, 10-22
members, 10-20
methods, 10-26
properties, 10-25
static methods, 10-24
OracleTruncateException Class, 10-19
OracleTypeException
constructors, 10-5
members, 10-3
properties, 10-8
static methods, 10-7
OracleTypeException Class, 10-2
OracleXmlPropertyCollection Class, 5-3
OracleXmlStream Class, 5-22
OracleXmlType, 3-37
OracleXmlType Class, 5-36

P
parameter binding, 3-10
password expiration, 3-6
performance, 3-21
array binding, 3-18
connection pooling, 3-3
fine-tuning FetchSize, 3-29
number of rows fetched, 3-29
Obtaining LOB Data, 3-27
PL/SQL Associative Array binding, 3-15
PL/SQL language, 3-30
PL/SQL REF CURSOR, 3-30
PL/SQL REF CURSOR and OracleRefCursor, 3-30
PLSQLAssociativeArray, 4-281
Pooling attribute, 3-3
populating an OracleDataReader from a REF CURSOR, 3-31
populating an OracleRefCursor from a REF CURSOR, 3-32
populating the DataSet from a REF CURSOR, 3-31
preventing data loss, 3-49, 3-50
PrimaryKey property, 3-51
configuring, 3-52
privileged connections, 3-5
proxy authentication, 3-7

R
readme.txt, 2-2
REF CURSOR
behavior of ExecuteScalar method for, 3-32
obtaining, 3-31
populating DataSet from, 3-31
populating from OracleDataReader, 3-31
Requery property, 3-51
round-trip, 3-18
RowSize property, 3-30

S
Safe Type Mapping, 3-49
SafeMapping Property, 3-50
Samples, 1-5, 2-2
saving changes
to Oracle 8i, 3-46
to Oracle9i, 3-47
using XML data, 3-44
SchemaTable, 4-153
session globalization parameters, 3-57
session globalization settings, 3-54
Size property, 3-15
special characters
in column data, 3-45
in table or view, 3-45
special characters in XML, 3-39
Statement Caching
  connection string attributes, 3-21
  methods and properties, 3-21
  Statement Cache Size, 3-21
  Statement Change Purge, 3-21
Stream class, 3-33
SYSDBA privileges, 3-5
SYSOPER privileges, 3-5
system requirements, 2-1
  Oracle Data Provider for .NET, 2-1

T
  table or view
    special characters in, 3-45
  TAF, 3-8
  TAF callback support, 3-8
  Temporary LOBs, 3-35
  thread globalization settings, 3-56
  thread-based globalization settings, 3-55
  TraceFileName, 3-57
  TraceLevel, 3-57
  TraceOption, 3-58
  Transaction object, 3-10
  Transparent Application Failover (TAF), 3-8
  troubleshooting, 3-57
  typed OracleDataReader accessors, 3-23

U
  unique columns, 3-27
  unique constraint, 3-27
  unique index, 3-27
  UniqueConstraint, 3-52
  uniqueness
    in updating DataSet to database, 3-51
    uniqueness in DataRows, 3-52
    updating
      LOBs, 3-34
    updating a DataSet Obtained from a REF CURSOR, 3-32
    updating LOBs using a DataSet, 3-34
    updating LOBs using ODP.NET LOB objects, 3-34
    updating LOBs using OracleCommand and
      OracleParameter, 3-34
    updating without PrimaryKey and Constraints, 3-53
    using FetchSize property, 3-29

V
  Value property, 3-16

X
  XML
    data manipulation using, 3-44
    special characters, 3-39
  XML data
    saving changes using, 3-44