## Contents

**Preface** ............................................................................................................................................................ xxix

- Audience ........................................................................................................................................................ xix
- Documentation Accessibility .......................................................................................................................... xix
- Related Documents .................................................................................................................................. xx
- Passwords in Code Examples ..................................................................................................................... xxi
- Conventions ............................................................................................................................................... xxi

**What’s New in Oracle Data Provider for .NET?** ................................................................................................................................. xxiii

- New Features in Oracle Data Provider for .NET Release 11.2.0.3 ........................................................ xxiii
- New Features in Oracle Data Provider for .NET Release 11.2.0.2 ........................................................ xxiv
- New Features in Oracle Data Provider for .NET Release 11.2.0.1.2 ................................................... xxv
- New Features in Oracle Data Provider for .NET Release 11.2 ............................................................... xxv
- New Features in Oracle Data Provider for .NET Release 11.1.0.7.20 ................................................ xxv
- New Features in Oracle Data Provider for .NET Release 11.1.0.6.20 ................................................ xxvii
- New Features in Oracle Data Provider for .NET Release 11.1 ............................................................ xxv
- New Features in Oracle Data Provider for .NET Release 10.2.0.4 ...................................................... xxv
- New Features in Oracle Data Provider for .NET Release 10.2.0.3 ...................................................... xxv
- New Features in Oracle Data Provider for .NET Release 10.2.0.2 ................................................... xxx
- New Features in Oracle Data Provider for .NET Release 10.2 ............................................................ xxx
- New Features in Oracle Data Provider for .NET Release 10.1.0.3 ................................................... xxxi
- New Features in Oracle Data Provider for .NET Release 10.1 ........................................................... xxxii
- New Features in Oracle Data Provider for .NET Release 9.2.0.4 ...................................................... xxxiii

**1 Introducing Oracle Data Provider for .NET**

- **.NET Data Access in Oracle: Products and Documentation** ................................................................. 1-1
  - Oracle Data Provider for .NET (ODP.NET) .......................................................................................... 1-1
  - Oracle Developer Tools for Visual Studio ......................................................................................... 1-2
  - Oracle Database Extensions for .NET ............................................................................................... 1-2
  - Oracle Providers for ASP.NET .......................................................................................................... 1-2
  - Oracle Services for Microsoft Transaction Server ........................................................................... 1-3
  - Oracle TimesTen In-Memory Database ........................................................................................... 1-3

- **Overview of Oracle Data Provider for .NET (ODP.NET)** ................................................................. 1-3
- **Oracle Data Provider for .NET Assembly** ......................................................................................... 1-4
  - Oracle.DataAccess.Client Namespace ............................................................................................ 1-4
  - Oracle.DataAccess.Types Namespace ............................................................................................ 1-9
# 2 Installing and Configuring
Oracle Data Provider for .NET

## System Requirements

- Oracle Data Provider for .NET Versioning Scheme
- Installing Oracle Data Provider for .NET
  - File Locations After Installation
  - Search Order for Unmanaged DLLs
- Configuring Oracle Data Provider for .NET
  - Supported Configuration Settings
  - Windows Registry
  - Configuration File Support
  - Sample Configuration Files
- Configuring a Port to Listen for Database Notifications
- General .NET Programming Recommendations and Tips for ODP.NET

## 3 Features of Oracle Data Provider for .NET

### Connecting to Oracle Database
- Connection String Attributes
- Specifying the Data Source Attribute
- Connection Pooling
- Connection Pool Management
- Connection Pool Performance Counters
- Edition-Based Redefinition
- Connecting to an Oracle Real Application Clusters (Oracle RAC) Database
- Operating System Authentication
- Privileged Connections
- Password Expiration
- Proxy Authentication
- Dynamic Distributed Transaction Enlistment
- Client Identifier and End-to-End Tracing
- Transparent Application Failover (TAF) Callback Support

### ADO.NET 2.0 Features
- About ADO.NET 2.0
- Base Classes and Provider Factory Classes
- Connection String Builder
- Data Source Enumerator
- Support for Code Access Security
- Support for Schema Discovery
- System.Transactions and Promotable Transactions
- Batch Processing Support
- ADO.NET 2.0 Only Classes and Class Members
- Bulk Copy Support

### ADO.NET Entity Framework and LINQ to Entities
- Mapping Oracle Data Types to EDM Types
- Oracle Number Default Data Type Mapping and Customization
OracleCommand Object ................................................................................................................................. 3-44
Transactions .................................................................................................................................................. 3-44
Parameter Binding ...................................................................................................................................... 3-45
Statement Caching ....................................................................................................................................... 3-56
Self-Tuning .................................................................................................................................................. 3-58
ODP.NET Types Overview ............................................................................................................................. 3-60
Obtaining Data from an OracleDataReader Object .......................................................................................... 3-62
Typed OracleDataReader Accessors ............................................................................................................... 3-62
Obtaining LONG and LONG RAW Data ....................................................................................................... 3-65
Obtaining LOB Data ...................................................................................................................................... 3-66
Controlling the Number of Rows Fetched in One Database Round-Trip ......................................................... 3-70
PL/SQL REF CURSOR and OracleRefCursor .................................................................................................. 3-72
Obtaining an OracleRefCursor Object ......................................................................................................... 3-72
Obtaining a REF CURSOR Data Type ............................................................................................................ 3-72
Populating an OracleDataReader from a REF CURSOR .................................................................................. 3-73
Populating the DataSet from a REF CURSOR ............................................................................................... 3-73
Populating an OracleRefCursor from a REF CURSOR .................................................................................. 3-73
Updating a DataSet Obtained from a REF CURSOR ..................................................................................... 3-74
Behavior of ExecuteScalar Method for REF CURSOR ................................................................................. 3-74
Passing a REF CURSOR to a Stored Procedure ............................................................................................ 3-74
Implicit REF CURSOR Binding ........................................................................................................................ 3-75
Specifying REF CURSOR Bind and Metadata Information in the .NET Configuration File ....................... 3-76
Sample Configuration File and Application ................................................................................................ 3-79
Usage Considerations ................................................................................................................................ 3-82
LOB Support ................................................................................................................................................. 3-83
Large Character and Large Binary Data Types ............................................................................................ 3-84
Oracle Data Provider for .NET LOB Objects .................................................................................................. 3-84
Updating LOBs Using a DataSet .................................................................................................................. 3-85
Updating LOBs Using OracleCommand and OracleParameter .................................................................... 3-85
Updating LOBs Using ODP.NET LOB Objects ............................................................................................. 3-86
Temporary LOBs ......................................................................................................................................... 3-86
ODP.NET XML Support ................................................................................................................................ 3-86
Supported XML Features ............................................................................................................................ 3-87
XQuery Support .............................................................................................................................................. 3-88
OracleXmlType and Connection Dependency .............................................................................................. 3-88
Updating XMLType Data in the Database .................................................................................................... 3-89
Updating XML Data in OracleXmlType ......................................................................................................... 3-90
Characters with Special Meaning in XML .................................................................................................... 3-90
Retrieving Query Result Set as XML ............................................................................................................ 3-91
Data Manipulation Using XML .................................................................................................................... 3-95
Oracle User-Defined Types (UDTs) and .NET Custom Types ...................................................................... 3-99
Oracle User-Defined Types (UDTs) ............................................................................................................... 3-100
Custom Types .............................................................................................................................................. 3-100
Specifying Custom Type Mappings ............................................................................................................ 3-103
Converting Between Custom Types and Oracle UDTs ................................................................................. 3-105
Oracle UDT Attribute Mappings .................................................................................................................. 3-106
Oracle UDT Retrieval from OracleDataReader .......................................................................................... 3-108
Oracle Data Provider for .NET Classes

Server-Side Features

OracleCommand Class

OracleCommand Members

OracleCommand Constructors

OracleCommand Static Methods

OracleCommand Properties

OracleCommand Public Methods

OracleCommandBuilder Class

OracleCommandBuilder Members

OracleCommandBuilder Constructors

OracleCommandBuilder Static Methods
OracleInfoMessageEventArgs Public Methods ................................................................. 5-246
OracleInfoMessageEventHandler Delegate ....................................................................... 5-247
OracleParameter Class ..................................................................................................... 5-248
  OracleParameter Members ............................................................................................... 5-250
  OracleParameter Constructors ......................................................................................... 5-252
  OracleParameter Static Methods ...................................................................................... 5-263
  OracleParameter Properties .............................................................................................. 5-264
  OracleParameter Public Methods ..................................................................................... 5-280
OracleParameterCollection Class ....................................................................................... 5-283
  OracleParameterCollection Members ............................................................................... 5-285
  OracleParameterCollection Static Methods ....................................................................... 5-287
  OracleParameterCollection Properties ............................................................................. 5-288
  OracleParameterCollection Public Methods ..................................................................... 5-291
OraclePermission Class .................................................................................................... 5-309
  OraclePermission Members .............................................................................................. 5-310
  OraclePermission Constructor ......................................................................................... 5-312
  OraclePermission Static Methods ................................................................................... 5-313
  OraclePermission Public Properties ................................................................................ 5-314
  OraclePermission Public Methods .................................................................................. 5-315
OraclePermissionAttribute Class ....................................................................................... 5-318
  OraclePermissionAttribute Members .............................................................................. 5-319
  OraclePermissionAttribute Constructor .......................................................................... 5-321
  OraclePermissionAttribute Static Methods ..................................................................... 5-322
  OraclePermissionAttribute Public Properties .................................................................. 5-323
  OraclePermissionAttribute Public Methods .................................................................. 5-324
OracleRowUpdatedEventArgs Class ............................................................................... 5-325
  OracleRowUpdatedEventArgs Members ............................................................................ 5-326
  OracleRowUpdatedEventArgs Constructor ....................................................................... 5-328
  OracleRowUpdatedEventArgs Static Methods .................................................................. 5-329
  OracleRowUpdatedEventArgs Properties ........................................................................ 5-330
  OracleRowUpdatedEventArgs Public Methods ............................................................... 5-331
OracleRowUpdatedEventHandler Delegate ........................................................................ 5-332
OracleRowUpdatingEventArgs Class ............................................................................... 5-333
  OracleRowUpdatingEventArgs Members ......................................................................... 5-334
  OracleRowUpdatingEventArgs Constructor ..................................................................... 5-336
  OracleRowUpdatingEventArgs Static Methods .................................................................. 5-337
  OracleRowUpdatingEventArgs Properties ....................................................................... 5-338
  OracleRowUpdatingEventArgs Public Methods .............................................................. 5-339
OracleRowUpdatingEventHandler Delegate ........................................................................ 5-340
OracleTransaction Class .................................................................................................. 5-341
  OracleTransaction Members ............................................................................................ 5-344
  OracleTransaction Static Methods .................................................................................. 5-345
  OracleTransaction Properties .......................................................................................... 5-346
  OracleTransaction Public Methods .................................................................................. 5-348
OracleConnectionType Enumeration ..................................................................................... 5-356
OracleCollectionType Enumeration .................................................................................... 5-357
OracleDBShutdownMode Enumeration .................................................................................. 5-358
OracleDBStartupMode Enumeration ............................................................... 5-359
OracleDbType Enumeration ............................................................................. 5-360
OracleParameterStatus Enumeration ............................................................. 5-362

6  Oracle Data Provider for .NET
XML-Related Classes

OracleXmlCommandType Enumeration ...................................................................... 6-2
OracleXmlQueryProperties Class ............................................................................ 6-3
OracleXmlQueryProperties Members ..................................................................... 6-7
OracleXmlQueryProperties Constructor .............................................................. 6-8
OracleXmlQueryProperties Properties ................................................................... 6-9
OracleXmlQueryProperties Public Methods ......................................................... 6-12
OracleXmlSaveProperties Class ............................................................................ 6-13
OracleXmlSaveProperties Members ..................................................................... 6-16
OracleXmlSaveProperties Constructor .............................................................. 6-17
OracleXmlSaveProperties Properties .................................................................... 6-18
OracleXmlSaveProperties Public Methods ......................................................... 6-22
OracleXmlStream Class ....................................................................................... 6-23
OracleXmlStream Members .................................................................................. 6-24
OracleXmlStream Constructor ............................................................................. 6-26
OracleXmlStream Static Methods ......................................................................... 6-27
OracleXmlStream Instance Properties ................................................................... 6-28
OracleXmlStream Instance Methods ..................................................................... 6-32
OracleXmlType Class ............................................................................................ 6-37
OracleXmlType Members ...................................................................................... 6-38
OracleXmlType Constructors ............................................................................... 6-40
OracleXmlType Static Methods ............................................................................ 6-43
OracleXmlType Instance Properties ...................................................................... 6-44
OracleXmlType Instance Methods ......................................................................... 6-49

7  ADO.NET 2.0 Classes

OracleConnectionFactory Class .............................................................................. 7-2
OracleConnectionFactory Members .......................................................................... 7-4
OracleConnectionFactory Public Properties ......................................................... 7-5
OracleConnectionFactory Public Methods ............................................................. 7-6
OracleConnectionStringBuilder Class .................................................................. 7-10
OracleConnectionStringBuilder Members ........................................................... 7-13
OracleConnectionStringBuilder Constructors .................................................... 7-16
OracleConnectionStringBuilder Public Properties ............................................. 7-18
OracleConnectionStringBuilder Public Methods ................................................ 7-33
OracleDataSourceEnumerator Class ...................................................................... 7-36
OracleDataSourceEnumerator Members .............................................................. 7-38
OracleDataSourceEnumerator Public Methods .................................................... 7-39

8  Oracle Data Provider for .NET HA Event Classes

OracleHAEventArgs Class .................................................................................... 8-2
9  Database Change Notification

OracleDependency Class ................................................................. 9-2
OracleDependency Members ......................................................... 9-3
OracleDependency Constructors .................................................... 9-5
OracleDependency Static Fields ..................................................... 9-9
OracleDependency Static Methods ............................................... 9-11
OracleDependency Properties ....................................................... 9-12
OracleDependency Methods ......................................................... 9-18
OracleDependency Events ............................................................. 9-21

OracleNotificationRequest Class .................................................. 9-22
OracleNotificationRequest Members ............................................. 9-23
OracleNotificationRequest Static Methods ................................. 9-24
OracleNotificationRequest Properties ........................................... 9-25
OracleNotificationRequest Methods .............................................. 9-29

OracleNotificationEventArgs Class .............................................. 9-30
OracleNotificationEventArgs Members ........................................ 9-31
OracleNotificationEventArgs Static Fields ................................. 9-32
OracleNotificationEventArgs Static Methods ............................. 9-33
OracleNotificationEventArgs Properties ...................................... 9-34
OracleNotificationEventArgs Methods ........................................ 9-39

OracleDependency Events ............................................................ 9-40

OracleGlobalization Class ............................................................. 9-2
OracleGlobalization Members ....................................................... 10-2
OracleGlobalization Static Methods .............................................. 10-4
OracleGlobalization Properties .................................................... 10-6
OracleGlobalization Public Methods ............................................ 10-12

10  Oracle Data Provider for .NET

Globalization Classes

OracleGlobalization Class ............................................................. 10-2
OracleGlobalization Members ....................................................... 10-4
OracleGlobalization Static Methods .............................................. 10-6
OracleGlobalization Properties .................................................... 10-12
OracleGlobalization Public Methods ............................................ 10-22

11  Oracle Data Provider for .NET

Failover Classes

OracleFailoverEventArgs Class ..................................................... 11-2
OracleFailoverEventArgs Members .............................................. 11-5
OracleFailoverEventArgs Static Methods ..................................... 11-6
OracleFailoverEventArgs Properties ............................................. 11-7
## 12 Advanced Queuing Support

- **OracleAQAgent Class**: 12-2
  - OracleAQAgent Members: 12-3
  - OracleAQAgent Constructors: 12-4
  - OracleAQAgent Properties: 12-6

- **OracleAQDequeueOptions Class**: 12-8
  - OracleAQDequeueOptions Members: 12-9
  - OracleAQDequeueOptions Constructor: 12-11
  - OracleAQDequeueOptions Properties: 12-12

- **OracleAQEnqueueOptions Class**: 12-19
  - OracleAQEnqueueOptions Members: 12-20
  - OracleAQEnqueueOptions Constructor: 12-21
  - OracleAQEnqueueOptions Properties: 12-22

- **OracleAQMessage Class**: 12-25
  - OracleAQMessage Members: 12-26
  - OracleAQMessage Constructors: 12-28
  - OracleAQMessage Properties: 12-30

- **OracleAQMessageAvailableEventArgs Class**: 12-38
  - OracleAQMessageAvailableEventArgs Members: 12-39
  - OracleAQMessageAvailableEventArgs Properties: 12-40

- **OracleAQQueue Class**: 12-47
  - OracleAQQueue Members: 12-48
  - OracleAQQueue Constructors: 12-49
  - OracleAQQueue Static Methods: 12-51
  - OracleAQQueue Properties: 12-58
  - OracleAQQueue Public Methods: 12-64
  - OracleAQQueue Events: 12-79

- **OracleAQDequeueMode Enumeration**: 12-84
- **OracleAQMessageDeliveryMode Enumeration**: 12-85
- **OracleAQMessageState Enumeration**: 12-87
- **OracleAQMessageType Enumeration**: 12-88
- **OracleAQNavigationMode Enumeration**: 12-89
- **OracleAQNotificationGroupingType Enumeration**: 12-90
- **OracleAQNotificationType Enumeration**: 12-91
- **OracleAQVisibilityMode Enumeration**: 12-92
13 Oracle Data Provider for .NET Types Classes

OracleBFile Class .................................................................................................................. 13-2
OracleBFile Members .............................................................................................................. 13-4
OracleBFile Constructors .................................................................................................... 13-7
OracleBFile Static Fields ..................................................................................................... 13-9
OracleBFile Static Methods ............................................................................................... 13-11
OracleBFile Instance Properties ....................................................................................... 13-12
OracleBFile Instance Methods ......................................................................................... 13-19

OracleBlob Class ................................................................................................................ 13-38
OracleBlob Members .......................................................................................................... 13-40
OracleBlob Constructors ................................................................................................... 13-43
OracleBlob Static Fields ..................................................................................................... 13-45
OracleBlob Static Methods .................................................................................................. 13-46
OracleBlob Instance Properties ......................................................................................... 13-47
OracleBlob Instance Methods ............................................................................................ 13-53

OracleClob Class ................................................................................................................ 13-73
OracleClob Members .......................................................................................................... 13-75
OracleClob Constructors ................................................................................................... 13-78
OracleClob Static Fields ..................................................................................................... 13-80
OracleClob Static Methods ............................................................................................... 13-81
OracleClob Instance Properties ......................................................................................... 13-82
OracleClob Instance Methods ............................................................................................ 13-88

OracleRefCursor Class ....................................................................................................... 13-113
OracleRefCursor Members .................................................................................................. 13-116
OracleRefCursor Static Methods ...................................................................................... 13-117
OracleRefCursor Properties ............................................................................................... 13-118
OracleRefCursor Instance Methods ................................................................................... 13-121

14 Oracle Data Provider for .NET Types Structures

OracleBinary Structure ....................................................................................................... 14-2
OracleBinary Members ...................................................................................................... 14-4
OracleBinary Constructor ................................................................................................. 14-7
OracleBinary Static Fields ................................................................................................. 14-8
OracleBinary Static Methods ............................................................................................. 14-9
OracleBinary Static Operators ......................................................................................... 14-15
OracleBinary Static Type Conversion Operators ............................................................. 14-21
OracleBinary Properties .................................................................................................... 14-23
OracleBinary Instance Methods ....................................................................................... 14-26

OracleDate Structure ........................................................................................................ 14-29
OracleDate Members .......................................................................................................... 14-31
OracleDate Constructors .................................................................................................... 14-34
OracleDate Static Fields ..................................................................................................... 14-39
OracleDate Static Methods ............................................................................................... 14-41
OracleDate Static Operators ............................................................................................ 14-47
OracleDate Static Type Conversions ............................................................................... 14-52
OracleDate Properties ....................................................................................................... 14-56
15 Oracle Data Provider for .NET Types Exceptions

OracleException Class ........................................................................................................ 15-2
  OracleException Members ................................................................................................. 15-3
  OracleException Constructors ......................................................................................... 15-5
  OracleException Static Methods ...................................................................................... 15-7
  OracleException Properties .............................................................................................. 15-8
  OracleException Methods ................................................................................................. 15-10
OracleNullValueException Class ......................................................................................... 15-11
  OracleNullValueException Members ................................................................................ 15-12
  OracleNullValueException Constructors ........................................................................ 15-14
  OracleNullValueException Static Methods .................................................................... 15-16
  OracleNullValueException Properties ............................................................................ 15-17
  OracleNullValueException Methods ................................................................................ 15-18
OracleTruncateException Class ............................................................................................ 15-19
  OracleTruncateException Members .................................................................................. 15-20
  OracleTruncateException Constructors .......................................................................... 15-22
  OracleTruncateException Static Methods ..................................................................... 15-24
  OracleTruncateException Properties ............................................................................. 15-25
  OracleTruncateException Methods .................................................................................. 15-26

16 Oracle Data Provider for .NET UDT-Related Classes

OracleCustomTypeMappingAttribute Class ........................................................................ 16-2
OracleCustomTypeMappingAttribute Members ................................................................. 16-4
OracleCustomTypeMappingAttribute Constructors ......................................................... 16-6
OracleCustomTypeMappingAttribute Static Methods ....................................................... 16-7
OracleCustomTypeMappingAttribute Properties ............................................................. 16-8
OracleCustomTypeMappingAttribute Attribute Methods .................................................. 16-9

OracleObjectMappingAttribute Class .............................................................................. 16-10
OracleObjectMappingAttribute Members ......................................................................... 16-11
OracleObjectMappingAttribute Constructors ................................................................. 16-13
OracleObjectMappingAttribute Static Methods ............................................................... 16-15
OracleObjectMappingAttribute Properties ....................................................................... 16-16
OracleObjectMappingAttribute Attribute Methods ......................................................... 16-18

OracleArrayMappingAttribute Class .............................................................................. 16-19
OracleArrayMappingAttribute Members ........................................................................... 16-20
OracleArrayMappingAttribute Constructors ................................................................. 16-22
OracleArrayMappingAttribute Static Methods ............................................................... 16-23
OracleArrayMappingAttribute Properties ....................................................................... 16-24
OracleArrayMappingAttribute Attribute Methods ......................................................... 16-25

IOracleCustomType Interface ............................................................................................ 16-26
IOracleCustomType Members ............................................................................................ 16-27
IOracleCustomType Interface Methods ............................................................................ 16-28

IOracleCustomTypeFactory Interface .................................................................................. 16-30
IOracleCustomTypeFactory Members ............................................................................... 16-31
IOracleCustomTypeFactory Interface Methods ................................................................. 16-32

IOracleArrayTypeFactory Interface .................................................................................... 16-33
IOracleArrayTypeFactory Members ................................................................................... 16-34
IOracleArrayTypeFactory Interface Methods ................................................................. 16-35

OracleUdt Class .................................................................................................................. 16-37
OracleUdt Members ........................................................................................................... 16-38
OracleUDT Static Methods ............................................................................................... 16-39

OracleRef Class .................................................................................................................. 16-51
OracleRef Members ........................................................................................................... 16-52
OracleRef Constructors ..................................................................................................... 16-54
OracleRef Static Fields ...................................................................................................... 16-56
OracleRef Static Methods .................................................................................................. 16-57
OracleRef Instance Properties .......................................................................................... 16-58
OracleRef Instance Methods ............................................................................................. 16-62

OracleUdtFetchOption Enumeration .................................................................................... 16-73
OracleUdtStatus Enumeration ........................................................................................... 16-74

17 Oracle Data Provider for .NET
Bulk Copy Classes

OracleBulkCopy Class ......................................................................................................... 17-2
OracleBulkCopy Members ................................................................................................. 17-3
OracleBulkCopy Constructors ............................................................................................. 17-5
OracleBulkCopy Properties ............................................................................................... 17-9
OracleBulkCopy Public Methods ....................................................................................... 17-14
OracleBulkCopy Events ..................................................................................................... 17-20
A Oracle Schema Collections

Common Schema Collections ................................................................. A-1
  MetaDataCollections ........................................................................ A-1
  DataSourceInformation .................................................................. A-2
  DataTypes ......................................................................................... A-3
  Restrictions ..................................................................................... A-4
  ReservedWords ................................................................................ A-5

ODP.NET-Specific Schema Collection .................................................. A-5
  Tables .................................................................................................. A-5
  Columns .............................................................................................. A-6
  Views ................................................................................................ A-6
  XMLSchema ..................................................................................... A-7
  Users ................................................................................................. A-7
  Synonyms ........................................................................................ A-7
  Sequences ......................................................................................... A-8
  Functions ........................................................................................ A-8
  Procedures ......................................................................................... A-9
  ProcedureParameters ....................................................................... A-10
  Arguments ......................................................................................... A-11
  Packages ........................................................................................ A-11
  PackageBodies ................................................................................ A-12
  JavaClasses ..................................................................................... A-13
  Indexes ............................................................................................. A-13
  IndexColumns .................................................................................. A-16
  PrimaryKeys .................................................................................... A-16
  ForeignKeys .................................................................................... A-17
  ForeignKeyColumns .......................................................................... A-18
  UniqueKeys ...................................................................................... A-18
  OracleBulkCopyColumnMapping Class .................................................. 17-21
  OracleBulkCopyColumnMapping Members ................................................. 17-22
  OracleBulkCopyColumnMapping Constructors ........................................ 17-23
  OracleBulkCopyColumnMapping Properties ............................................. 17-26
  OracleBulkCopyColumnMappingCollection Class ........................................ 17-29
  OracleBulkCopyColumnMappingCollection Members ................................... 17-31
  OracleBulkCopyColumnMappingCollection Properties ................................ 17-32
  OracleBulkCopyColumnMappingCollection Public Methods ......................... 17-33
  OracleBulkCopyOptions Enumeration .................................................... 17-42
  OracleRowsCopiedEventHandler Delegate ............................................... 17-43
  OracleRowsCopiedEventArgs Class ....................................................... 17-44
    OracleRowsCopiedEventArgs Members .................................................. 17-45
    OracleRowsCopiedEventArgs Constructors ............................................ 17-46
    OracleRowsCopiedEventArgs Properties ............................................... 17-47
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 Preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Passwords in Code Examples
- 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 .NET, or another .NET language.

Although the examples in the documentation and the samples in the sample directory are written in C#, developers can use these examples as models for writing code in other .NET languages.

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

**Documentation Accessibility**


**Access to Oracle Support**

Oracle customers have access to electronic support through My Oracle Support. For information, visit
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 Advanced Application Developer’s Guide
- Oracle Database Application Developer’s Guide - Large Objects
- Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide
- Oracle Database New Features
- Oracle Database Concepts
- Oracle Database Reference
- Oracle Database Extensions for .NET Developer’s Guide
- Oracle Database Application Developer’s Guide - Object-Relational Features
- 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 Database Globalization Support Guide
- Oracle XML DB Developer’s Guide
- Oracle XML Developer’s Kit Programmer’s Guide
- Oracle Database Security Guide
- Oracle Spatial User’s Guide and Reference
- Oracle Data Guard Concepts and Administration

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
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://www.oracle.com/technology/documentation/index.html

For additional information, see:
http://msdn.microsoft.com/netframework

and
http://msdn.microsoft.com/library

**Passwords in Code Examples**

For simplicity in demonstrating this product, code examples do not perform the password management techniques that a deployed system normally uses. In a production environment, follow the Oracle Database password management guidelines, and disable any sample accounts. See *Oracle Database Security Guide* for password management guidelines and other security recommendations.

**Conventions**

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><code>monospace</code></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
What's New in Oracle Data Provider for .NET?

This section describes new features in Oracle Data Provider for .NET and provides references 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 11.2.0.3
- New Features in Oracle Data Provider for .NET Release 11.2.0.2
- New Features in Oracle Data Provider for .NET Release 11.2.0.1.2
- New Features in Oracle Data Provider for .NET Release 11.2
- New Features in Oracle Data Provider for .NET Release 11.1.0.7.20
- New Features in Oracle Data Provider for .NET Release 11.1.0.6.20
- New Features in Oracle Data Provider for .NET Release 11.1
- New Features in Oracle Data Provider for .NET Release 11.0.4
- New Features in Oracle Data Provider for .NET Release 10.2.0.3
- New Features in Oracle Data Provider for .NET Release 10.2.0.2
- New Features in Oracle Data Provider for .NET Release 10.2
- 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 11.2.0.3**

Oracle Data Provider for .NET release 11.2.0.3 includes the following:

- ADO.NET Entity Framework and LINQ to Entities Support

  ODP.NET now includes support for the ADO.NET Entity Framework and LINQ to Entities. Entity Framework is a framework for providing object-relational mapping service on data models. Entity Framework addresses the impedance mismatch between the relational database format and the client’s preferred object format. Language Integrated Query (LINQ) defines a set of operators that can be used to query, project, and filter data in arrays, enumerable classes, XML, relational databases, and other data sources. One form of LINQ, LINQ to Entities,
allows querying of Entity Framework data sources. ODP.NET supports Entity Framework such that the Oracle database can participate in object-relational modeling and LINQ to Entities queries.

Entity Framework and LINQ provides productivity benefits for the .NET developer. It abstracts the database's data model from the application's data model. Working with object-relational data becomes easier with Entity Framework's tools. Oracle's integration with Entity Framework and LINQ enables Oracle .NET developers to take advantage of all these productivity benefits.

See “ADO.NET Entity Framework and LINQ to Entities” on page 3-32 for more information on ODP.NET support for the ADO.NET Entity Framework and LINQ to Entities.

■ WCF Data Services and OData

Windows Communication Foundation (WCF) Data Services enable developers to create services that use the Open Data Protocol (OData) to expose and consume data over the internet by using the semantics of representational state transfer (REST). OData exposes data as resources that are addressable by URIs. OData uses Entity Data Model conventions to expose resources as sets of entities that are related by associations. ODP.NET supports Entity Framework, and can expose its data through OData and WCF Data Services.

WCF Data Services and OData facilitate creating flexible data services from any data source and naturally integrating them with the Web. All data sources, including Oracle databases, can be used by the same data sharing standard making data exchange more interoperable.

■ Implicit REF CURSOR Parameter Binding

ODP.NET can bind REF CURSOR parameters for stored procedures without binding them explicitly. To do so, the application must provide the REF CURSOR metadata as part of the .NET configuration file. This feature allows Entity Framework Function Import to call Oracle stored procedures and return REF CURSOR result sets. ODP.NET can also update the database's data with a DataSet or DataTable obtained through a REF CURSOR.

In Entity Framework, result set parameters are generally not declared. By supporting the implicit REF CURSOR parameter, ODP.NET more closely integrates with typical Entity Framework usage scenarios.

See “Implicit REF CURSOR Binding” on page 3-75 for detailed information on implicit REF CURSOR parameter binding.

New Features in Oracle Data Provider for .NET Release 11.2.0.2

Oracle Data Provider for .NET release 11.2.0.2 includes the following:

■ 64-bit ODP.NET XCopy for Windows x64

Now available for Windows x64 systems, ODP.NET XCopy provides system administrators with a smaller client install size than the standard ODP.NET client, and is easier to configure. ODP.NET XCopy simplifies embedding ODP.NET in customized deployment packages.

See Also:  XCopy under "Installing Oracle Data Provider for .NET" on page 2-3

■ TimesTen In-Memory Database Support
Oracle Data Provider for .NET enables fast data access for any .NET application, such as C# .NET, Visual Basic .NET, and ASP.NET, to TimesTen In-memory databases. ODP.NET support for TimesTen includes the classes, enumerations, interfaces, delegates and structures of the Oracle.DataAccess.Client and Oracle.DataAccess.Types namespaces. ODP.NET supports TimesTen Release 11.2.1.6.1 or later on Microsoft Windows 32-bit and 64-bit platforms. TimesTen can be used with .NET Framework 2.0, 3.0, 3.5, and 4 with Microsoft Visual Studio 2005 or later.

See Also: The latest TimesTen In-Memory Database documentation and resources can be accessed from:


New Features in Oracle Data Provider for .NET Release 11.2.0.1.2

Oracle Data Provider for .NET release 11.2.0.1.2 includes the following:

- Support for Microsoft .NET Framework 4

  ODP.NET for .NET Framework 4 supports .NET Framework 4 and the .NET Framework 4 Client Profile.

New Features in Oracle Data Provider for .NET Release 11.2

Oracle Data Provider for .NET release 11.2 includes the following:

- End-to-End Tracing: ClientInfo Property

  ODP.NET now supports the ClientInfo write-only property, in addition to the ActionName, ClientId, and ModuleName properties, on the OracleConnection object. This property specifies the client information for the connection.

  The ClientInfo property is an end-to-end tracing attribute that can be set on the client or middle tier. This attribute is propagated to the database server whenever the next server round-trip happens. This reduces the added overhead associated with an independent database round trip. Using the ClientInfo property is helpful in tracking database user activities and debugging applications.

  See Also:

  - "Client Identifier and End-to-End Tracing" on page 3-17
  - "ClientInfo" on page 5-78

- Edition-Based Redefinition

  Edition-based redefinition enables you to upgrade the database component of an application even while the .NET application is being used. This minimizes or eliminates downtime for the application.

  See Also: "Edition-Based Redefinition" on page 3-9

New Features in Oracle Data Provider for .NET Release 11.1.0.7.20

Oracle Data Provider for .NET release 11.1.0.7.20 includes the following:

- Self-Tuning for Applications
Based on run-time sampling, ODP.NET dynamically adjusts statement cache size to provide better application performance. Self-tuning also takes into account memory usage on the client machine in order to prevent excessive memory usage. Self-tuning improves ODP.NET performance, reduces network usage, and decreases server CPU and client CPU activity.

**See Also:** "Self-Tuning" on page 3-58

**Faster Data Retrieval and Less Memory Usage**

Retrieving data using `OracleDataReader` or populating a `DataSet` from an `OracleDataAdapter` is now faster.

ODP.NET reuses the same fetch array buffer for statements executed non-concurrently, saving on memory usage. The fetch array buffer stores data retrieved from the database.

No code changes are necessary to use these features. These features provide better performance and scalability for ODP.NET applications.

**Oracle Streams Advanced Queuing Support**

ODP.NET supports access to Oracle Streams Advanced Queuing (AQ). AQ provides database-integrated message queuing functionality to store messages persistently, propagate messages between queues on different machines and databases, and transmit messages using Oracle NET services, HTTP, HTTPS and SMTP.

ODP.NET can access all the operational features of AQ, such as enqueue, dequeue, listen and notification. Oracle Developer Tools for Visual Studio can administer and manage AQ resources.

**See Also:** "Oracle Streams Advanced Queuing Support" on page 3-115

**Promotable Local Transaction Support**

Distributed transactions require the orchestration of application, transaction coordinator, and multiple databases. Local transactions only require an application and a single resource manager, or database. Local transactions have less of an overhead when compared to distributed transactions.

It may be difficult to determine whether a transaction will be local or distributed at design time. Developers are forced to design applications for distributed transactions, even if local transactions are used most of the time. This situation leads to more resource usage than necessary at run time.

Promotable local transactions allow all transactions to remain local until more than one database is brought into the transaction. At this point, the transaction is promoted to a distributed transaction so that it can be managed by the transaction coordinator. This provides a better utilization of system resources. This feature is supported with Oracle Database 11g release 1 (11.1.0.7) and higher.

**See Also:** "System.Transactions and Promotable Transactions" on page 3-25

**ODP.NET Security Enhancements**
ODP.NET makes use of the OraclePermission class to enforce imperative security. This helps ensure that a user or application has a security level adequate for accessing data.

**See Also:**
- "Support for Code Access Security" on page 3-22
- "OraclePermission Class" on page 5-309
- "OraclePermissionAttribute Class" on page 5-318

**Callbacks for HA Event Notifications**

ODP.NET can register for Oracle High Availability (HA) events when "ha events=true" is specified in the connection string. ODP.NET is then able to receive notifications on which database, service, host, or instance has gone down or come up. .NET developers can register a callback with ODP.NET to notify the application when one of these events occurs and subsequently execute an event handler, as needed.

**See Also:**
- Chapter 8, "Oracle Data Provider for .NET HA Event Classes" on page 8-1
- "HAEvent" on page 5-113
- "OracleConnection Properties" on page 5-76

**Database Startup and Shutdown Operations**

Users with database administrator privileges can use the OracleDatabase class to startup or shutdown a database instance.

**See Also:**
- "OracleDatabase Class" on page 5-140
- "Shutdown" on page 5-145
- "Startup" on page 5-149

### New Features in Oracle Data Provider for .NET Release 11.1.0.6.20

Oracle Data Provider for .NET release 11.1.0.6.20 includes the following:

**32-bit ODP.NET XCopy**

Oracle XCopy provides system administrators with an ODP.NET client that is smaller in disk size than the standard ODP.NET client and is easily configurable. Oracle XCopy makes embedding ODP.NET in customized deployment packages much simpler.

**See Also:**  "XCopy" on page 2-3

**Support for Oracle User-Defined Types**

ODP.NET has the ability to represent Oracle UDTs defined in the database as custom types in .NET applications.
See Also:
- "Oracle User-Defined Types (UDTs) and .NET Custom Types" on page 3-99
- Chapter 16, "Oracle Data Provider for .NET UDT-Related Classes" on page 16-1

Bulk Copy Operations
ODP.NET supports the Bulk Copy operations to load a large amount of data efficiently.

See Also:
- "Bulk Copy Support" on page 3-30
- Chapter 17, "Oracle Data Provider for .NET Bulk Copy Classes" on page 17-1

Additional Connection Pool Optimizations for Oracle Real Application Clusters (Oracle RAC) and Oracle Data Guard
ODP.NET now cleans up the connection pool when the database down event is received from Oracle RAC or Oracle Data Guard. This is in addition to the events that ODP.NET already cleaned up the connection pool for: node down, service member down, and service down events.

See Also: "Connection Optimization for Oracle RAC and Oracle Data Guard" on page 3-10

Windows-Authenticated User Connection Pooling
Operating system-authenticated connections can now be managed as part of ODP.NET connection pools

See Also: "Operating System Authentication" on page 3-12

Connection Pool Performance Counters
ODP.NET publishes performance counters for connection pooling, which can be viewed using the Windows Performance Monitor.

See Also: "Connection Pool Performance Counters" on page 3-7

End-to-End Tracing Attribute Support
ODP.NET supports the ActionName, ClientId, ClientInfo, and ModuleName write-only properties on the OracleConnection object. These properties correspond to end-to-end tracing attributes that can be set on the client or middle-tier, and propagated to the database server whenever the next server round-trip happens. This reduces the added overhead associated with an independent database round trip. Using these attributes is helpful in tracking database user activities and debugging applications.

See Also: "Client Identifier and End-to-End Tracing" on page 3-17
New Features in Oracle Data Provider for .NET Release 11.1

Oracle Data Provider for .NET release 11.1 includes the following:

- Performance Enhancements

  The following performance enhancements have been made:

  - Improved Parameter Context Caching

    This release enhances the existing caching infrastructure to cache ODP.NET parameter contexts. This enhancement is independent of database version and it is available for all the supported database versions. This feature provides significant performance improvement for the applications that execute the same statement repeatedly.

    This enhancement is transparent to the developer. No code changes are needed to use this feature.

  - Efficient LOB Retrieval with LOBS or SecureFiles

    When using LOBS or SecureFiles, this release improves the performance of small-sized LOB retrieval by reducing the number of round-trips to the database. SecureFiles is available with Oracle 11g release 1 or later database versions.

    This enhancement is transparent to the developer. No code changes are needed to use this feature.

New Features in Oracle Data Provider for .NET Release 10.2.0.4

Oracle Data Provider for .NET release 10.2.0.4 includes the following:

- ODP.NET Configuration

  Developers can now configure ODP.NET using configuration files, including application.config, web.config, or machine.config.

  Settings in the machine.config override the registry settings. The settings in the application.config or the web.config override the values in the machine.config.

  See Also: "Configuring Oracle Data Provider for .NET" on page 2-5

New Features in Oracle Data Provider for .NET Release 10.2.0.3

Oracle Data Provider for .NET release 10.2.0.3 includes the following:

- 64-bit ODP.NET for Windows x64 and Windows Itanium

  ODP.NET natively supports the 64-bit .NET Framework for both 64-bit Windows platforms:

  - Windows x64 for AMD64 and Intel EM64T processors
  - 64-bit Windows for Intel Itanium

  64-bit systems allow for more scalable and better performing ODP.NET applications.

- Configuring FetchSize Through the Windows Registry

  This feature enables applications to specify the default result set fetch size through the registry.
Local Transaction Support for System.Transactions
This feature enables System.Transactions to use local transactions rather than distributed transactions. This can be specified either through the registry or through a connection string attribute.

See Also: "Local Transaction Support for Older Databases" on page 3-28

New Features in Oracle Data Provider for .NET Release 10.2.0.2
Oracle Data Provider for .NET release 10.2.0.2 includes the following:

- Support for Microsoft ADO.NET 2.0, including.
  - Provider Factory Classes and Base Classes
    Simplifies data access code to access multiple data sources with a provider generic API.
  - Connection String Builder
    Makes creating connections strings less error-prone and easier to manage.
  - Data Source Enumerator
    Enables the application to generically obtain a collection of the Oracle data sources that the application can connect to.
  - Support for Schema Discovery
    Permits application developers to find and return database schema information, such as tables, columns, and stored procedures.
  - System.Transactions Support
    ODP.NET supports implicit and explicit transactions using the System.Transactions namespace models.
  - Batch Processing Support
    Enables batch processing when the OracleDataAdapter.Update method is called.

See Also: "ADO.NET 2.0 Features" on page 3-20

New Features in Oracle Data Provider for .NET Release 10.2
Oracle Data Provider for .NET release 10.2 includes the following:

- Server-Side Features
  Server-side features for Oracle Data Provider for .NET provide data access from .NET stored procedures. Such procedures are enabled by Oracle Database Extensions for .NET, a new feature included with Oracle database on Windows.

See Also:
- Chapter 4, "Oracle Data Provider for .NET Server-Side Features"
- Oracle Database Extensions for .NET Developer’s Guide

- Support for Client Identifier
Oracle Data Provider for .NET exposes the `OracleConnection.ClientId` property, thus providing support for Oracle Virtual Private Database (VPD) and application context. Client identifier makes configuring VPD simpler for the developer.

See Also:  "Client Identifier and End-to-End Tracing" on page 3-17

- **Connection Pool Optimizations for Oracle Real Application Clusters (Oracle RAC)**

  Oracle Data Provider for .NET optimizes connection pooling for Oracle RAC databases by balancing work requests across Oracle RAC instances, based on the load balancing advisory and service goal. Furthermore, the ODP.NET connection pool can be enabled to proactively free resources associated with connections that have been severed due to a down Oracle RAC service, instance, or node.

  See Also:  "Connecting to an Oracle Real Application Clusters (Oracle RAC) Database" on page 3-10

- **Database Change Notification Support**

  Oracle Data Provider for .NET provides a notification framework that supports Continuous Query Notification. This enables applications to receive notifications when there is a change in a query result set or a change in the state of the database.

  See Also:
  - "Database Change Notification Support" on page 3-120
  - Chapter 9, "Database Change Notification"

- **Connection Pooling Management**

  Oracle Data Provider for .NET connection pool management provides explicit connection pool control to ODP.NET applications. Applications can explicitly clear connections in a connection pool or all the connection pools.

  See Also:  "Connection Pool Management" on page 3-7

- **Better LOB performance and functionality with Oracle Database 10g release 2 (10.2) and later**

  See Also:  "InitialLOBFetchSize" on page 5-18

- **Support for IN and IN/OUT REF CURSOR Objects**

  This feature enables applications to retrieve REF Cursors from a PL/SQL procedure or function and pass them to another stored procedure or function.

  See Also:  "Passing a REF CURSOR to a Stored Procedure" on page 3-74

### 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 3-56

- .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 5-96
  - "HasRows" on page 5-162

- Support for Command Cancellation and Timeout
  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.

  See Also:
  - "CommandTimeout" on page 5-15
  - "Cancel" on page 5-27

- DeriveParameters 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: "DeriveParameters" on page 5-49

- LOB Retrieval Enhancement
  Entire LOB column data can be retrieved even if the select list does not contain a primary key, ROWID, or unique key. This enhancement is available by setting the InitialLOBFetchSize property value to -1 for CLOB and BLOB objects.

  See Also: "Setting InitialLOBFetchSize to -1" on page 3-69

- LONG Retrieval Enhancement
  Entire LONG column data can be retrieved even if the select list does not contain a primary key, ROWID, or unique key. This enhancement is available by setting the InitialLONGFetchSize property value to -1.

  See Also: "Setting InitialLONGFetchSize to -1" on page 3-66

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 data types in the database

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

**See Also:** "Data Types BINARY_FLOAT and BINARY_DOUBLE" on page 3-45

- 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 HOMEID 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 as Oracle Database native type, XMLType.

- Access relational and object-relational data as XML data from an Oracle Database instance into a Microsoft .NET environment, process the XML using the Microsoft .NET Framework.

- Save changes to the database using XML data.

  **See Also:** "OPD.NET XML Support" on page 3-86

- 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 Binding" on page 3-50

- Support for InitialLOBFetchSize property on OracleCommand and OracleDataReader objects

  **See Also:** "Obtaining LOB Data" on page 3-66
This chapter introduces Oracle Data Provider for .NET (ODP.NET), an implementation of a .NET data provider for Oracle Database.

This chapter contains these topics:

- .NET Data Access in Oracle: Products and Documentation
- Overview of Oracle Data Provider for .NET (ODP.NET)
- Oracle Data Provider for .NET Assembly
- Using ODP.NET Client Provider in a Simple Application

**.NET Data Access in Oracle: Products and Documentation**

This section discusses Oracle components and products that work together to provide .NET data access to Oracle Database, how they relate to each other, and what documentation is provided.

These Oracle products provide .NET integration on the Windows operating system:

**Oracle Data Provider for .NET (ODP.NET)**

Oracle Data Provider for .NET provides fast data access from .NET clients to Oracle databases. ODP.NET enables .NET applications to take advantage of Oracle advanced features, such as Oracle Real Application Clusters (Oracle RAC) and XML DB. It is accessible through any .NET language, including C#, Visual Basic .NET, and C++ .NET.

*Oracle Data Provider for .NET Developer’s Guide* describes Oracle Data Provider for .NET features, their use, installation, requirements, and classes. The guide distinguishes which classes are supported in .NET stored procedures and which classes are supported for .NET clients.

Additionally, Oracle Data Provider for .NET Dynamic Help, which is context-sensitive online help, contains the same reference sections available in *Oracle Data Provider for .NET Developer’s Guide*, this guide.

Oracle Data Provider for .NET Dynamic Help is integrated with Visual Studio Dynamic Help. With Dynamic Help, you can access Oracle Data Provider for .NET documentation within Visual Studio by placing the cursor on an Oracle Data Provider for .NET keyword and pressing the F1 function key.
Oracle Developer Tools for Visual Studio

Oracle Developer Tools is an add-in to Visual Studio that provides graphical user interface (GUI) access to Oracle functionality. It provides improved developer productivity and ease of use. Oracle Developer Tools provide the ability to build .NET stored procedures using Visual Basic .NET, C#, and other .NET languages.

Oracle Developer Tools for Visual Studio Help describes Oracle Developer Tools. This help is in the form of dynamic help, which installs as part of the product. Additionally, the Oracle Developer Tools for Visual Studio Help includes the following documentation:

- Oracle Database PL/SQL User’s Guide and Reference
- Oracle Database SQL Reference
- Oracle Database Extensions for .NET Developer’s Guide
- Oracle Database Error Messages
- Access to Oracle Data Provider for .NET Dynamic Help
- Access to Oracle Providers for ASP.NET Dynamic Help

Oracle Database Extensions for .NET

Oracle Database Extensions for .NET provides the following:

- Hosting of Microsoft Common Language Runtime (CLR) in an external process on the server side, to execute .NET stored procedures.
- ODP.NET data access on the server side, from within the .NET stored procedure.

Oracle Database Extensions for .NET features, their use, installation, and requirements are described in Oracle Database Extensions for .NET Developer’s Guide.

Oracle Data Provider for .NET Developer’s Guide describes all ODP.NET classes. Classes that are not supported by Oracle Database Extensions for .NET are described as Not Supported in a .NET Stored Procedure.

See Also:

- Oracle Developer Tools for Visual Studio Help
- Oracle Database Extensions for .NET Developer’s Guide for more information about .NET stored procedures and functions
- "Oracle Data Provider for .NET Assembly" on page 1-4 for class listings
- Chapter 4, "Oracle Data Provider for .NET Server-Side Features"

Oracle Providers for ASP.NET

Oracle Providers for ASP.NET offer ASP.NET developers an easy to use method to store state common to web applications within an Oracle database. These providers are modeled on existing Microsoft ASP.NET providers, sharing similar schema and programming interfaces to provide .NET developers a familiar interface. Oracle supports the following providers:

- Cache Dependency Provider
- Membership Provider
- Profile Provider
Overview of Oracle Data Provider for .NET (ODP.NET)

Introducing Oracle Data Provider for .NET

- Role Provider
- Session State Provider
- Site Map Provider
- Web Events Provider
- Web Parts Personalization Provider

Oracle Providers for ASP.NET classes, their use, installation, and requirements are described in Oracle Providers for ASP.NET Developer’s Guide, which is also provided as dynamic help.

Oracle Services for Microsoft Transaction Server

Oracle Services for Microsoft Transaction Server (OraMTS) permit Oracle databases to be used as resource managers in Microsoft application coordinated transactions. OraMTS acts as a proxy for the Oracle database to the Microsoft Distributed Transaction Coordinator (MSDTC). As a result, OraMTS provides client-side connection pooling and allows client components that leverage Oracle to participate in promoteable and distributed transactions. In addition, OraMTS can operate with Oracle databases running on any operating system, given that the services themselves are run on Windows.

Oracle Services for Microsoft Transaction Server Developer’s Guide describes OraMTS, which allows Oracle databases to be used as resource managers in distributed transactions.

Oracle TimesTen In-Memory Database

ODP.NET support for Oracle TimesTen In-Memory Database (TimesTen) provides fast and efficient ADO.NET data access for applications that require the highest performance.

You can use ODP.NET with any of the following TimesTen installations:
- TimesTen Data Manager only (for direct connections)
- TimesTen Client only (for client/server connections, assuming a TimesTen Data Manager instance and TimesTen Server instance are accessible elsewhere)
- TimesTen Data Manager with TimesTen Server

For more information on ODP.NET features specific to a TimesTen environment, refer to the Oracle Data Provider for .NET Oracle TimesTen In-Memory Database Support User’s Guide.

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

Oracle Data Provider for .NET (ODP.NET) is an implementation of a .NET data provider for Oracle Database, using and inheriting from classes and interfaces available in the Microsoft .NET Framework Class Library.

Following the .NET Framework, ODP.NET uses the ADO.NET model, which allows native providers to expose provider-specific features and data types. This is similar to Oracle Provider for OLE DB, where 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.
Oracle Data Provider for .NET uses Oracle native APIs to offer fast and reliable access to Oracle data and features from any .NET application.

The ODP.NET classes described in this guide are contained in the Oracle.DataAccess.dll assembly.

- **Client Applications:** All ODP.NET classes are available for use in client applications.
- **.NET Stored Procedures:** Most ODP.NET classes can be used from within .NET stored procedures and functions. Those classes which cannot, are labeled *Not Supported in a .NET Stored Procedure*. Additionally, some classes contain members which may not be supported, and this is so indicated in the member tables that follow the class descriptions, and listed in Chapter 4 of this guide.

See Also:
- Table 4–1, "API Support Comparison Between Client Application and .NET Stored Procedure"
- "Oracle Data Provider for .NET Assembly" on page 1-4 for class lists
- Chapter 4, "Oracle Data Provider for .NET Server-Side Features"
- *Oracle Database Extensions for .NET Developer’s Guide* for more information about .NET stored procedures and functions

### Oracle Data Provider for .NET Assembly

The Oracle.DataAccess.dll assembly provides two namespaces:

- The *Oracle.DataAccess.Client* namespace contains ODP.NET classes and enumerations for the client-side provider.
- The *Oracle.DataAccess.Types* namespace contains the Oracle Data Provider for .NET data types (ODP.NET Types).

### Oracle.DataAccess.Client Namespace

The *Oracle.DataAccess.Client* namespace contains implementations of core ADO.NET classes and enumerations for ODP.NET, as well as ODP.NET specific classes.

The following tables list ODP.NET classes, enumerations, and types that are supported by the *Oracle.DataAccess.Client* namespace. The tables also indicated which classes are not supported in .NET stored procedures.

#### Oracle.DataAccess.Client

*Table 1–1 Oracle.DataAccess.Client*

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
</table>
| OnChangeEventHandler Delegate | The OnChangedEventHandler event delegate represents the signature of the method that handles the notification.  
**Not Supported in a .NET Stored Procedure** |
| OracleAQAgent Class       | The OracleAQAgent class represents agents that may be senders or recipients of a message. |
OracleAQDequeueOptions Class
An OracleAQDequeueOptions object represents the options available when dequeuing a message from an OracleAQQueue object.

OracleAQEnqueueOptions Class
The OracleAQEnqueueOptions class represents the options available when enqueuing a message to an OracleAQQueue.

OracleAQMessage Class
An OracleAQMessage object represents a message to be enqueued and dequeued.

OracleAQMessageAvailableEventArgs Class
The OracleAQMessageAvailableEventArgs class provides event data for the OracleAQQueue.MessageAvailable event.

OracleAQMessageAvailableEventHandler Delegate
The OracleAQMessageAvailableEventHandler delegate represents the signature of the method that handles the OracleAQQueue.MessageAvailable event.

OracleAQQueue Class
An OracleAQQueue object represents a queue.

OracleBulkCopy Class
An OracleBulkCopy object efficiently bulk loads or copies data into an Oracle table from another data source.

OracleBulkCopyColumnMapping Class
The OracleBulkCopyColumnMapping class defines the mapping between a column in the data source and a column in the destination database table.

OracleBulkCopyColumnMappingCollection Class
The OracleBulkCopyColumnMappingCollection class represents a collection of OracleBulkCopyColumnMapping objects that are used to map columns in the data source to columns in a destination table.

OracleClientFactory Class
An OracleClientFactory object allows applications to instantiate ODP.NET classes in a generic way.

OracleCommand Class
An OracleCommand object represents a SQL command, a stored procedure or function, or a table name.

OracleCommandBuilder Class
An OracleCommandBuilder object provides automatic SQL generation for the OracleDataAdapter when the database is updated.

OracleConnection Class
An OracleConnection object represents a connection to Oracle Database.

OracleConnectionStringBuilder Class
An OracleConnectionStringBuilder object allows applications to create or modify connection strings.

OracleDataAdapter Class
An OracleDataAdapter object represents a data provider object that communicates with the DataSet.

OracleDataReader Class
An OracleDataReader object represents a forward-only, read-only, in-memory result set.
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleDataSourceEnumerator Class</td>
<td>An OracleDataSourceEnumerator object allows applications to generically obtain a collection of data sources to connect to.</td>
</tr>
<tr>
<td>OracleDependency Class</td>
<td>An OracleDependency class represents a dependency between an application and an Oracle database.</td>
</tr>
<tr>
<td>OracleDependency Class</td>
<td>Not Supported in a .NET Stored Procedure</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 class provides event data for the OracleConnection.Failover event.</td>
</tr>
<tr>
<td>OracleFailoverEventArgs Class</td>
<td>Not Supported in a .NET Stored Procedure</td>
</tr>
<tr>
<td>OracleFailoverEventHandler Delegate</td>
<td>The OracleFailoverEventHandler represents the signature of the method that handles the OracleConnection.Failover event.</td>
</tr>
<tr>
<td>OracleFailoverEventArgs Class</td>
<td>Not Supported in a .NET Stored Procedure</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>OracleHAEventArgs Class</td>
<td>The OracleHAEventArgs class provides event data for the OracleConnection.HAEvent event.</td>
</tr>
<tr>
<td>OracleHAEventHandler Delegate</td>
<td>The OracleHAEventHandler delegate represents the signature of the method that handles the OracleConnection.HAEvent event.</td>
</tr>
<tr>
<td>OracleInfoMessageEventArgs Class</td>
<td>The OracleInfoMessageEventArgs object provides event data for the OracleConnection.InfoMessage event.</td>
</tr>
<tr>
<td>OracleInfoMessageEventArgs Class</td>
<td>The OracleInfoMessageEventArgs class 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>OracleNotificationEventArgs Class</td>
<td>The OracleNotificationEventArgs class provides event data for a notification.</td>
</tr>
<tr>
<td>OracleNotificationRequest Class</td>
<td>An OracleNotificationRequest class represents a notification request to be subscribed in the database.</td>
</tr>
<tr>
<td>OracleNotificationRequest Class</td>
<td>Not Supported in a .NET Stored Procedure</td>
</tr>
<tr>
<td>OracleParameter Class</td>
<td>An OracleParameter object represents a parameter for an OracleCommand.</td>
</tr>
</tbody>
</table>
### Oracle.DataAccess.Client Enumerations

Table 1–2 lists the client enumerations.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OracleParameterCollection Class</strong></td>
<td>An OracleParameterCollection object represents a collection of OracleParameters.</td>
</tr>
<tr>
<td><strong>OraclePermission Class</strong></td>
<td>An OraclePermission object enables ODP.NET to enforce imperative security and helps ensure that a user has a security level adequate for accessing data.</td>
</tr>
<tr>
<td><strong>OraclePermissionAttribute Class</strong></td>
<td>An OraclePermissionAttribute object enables ODP.NET to enforce declarative security and helps ensure that a user has a security level adequate for accessing data.</td>
</tr>
<tr>
<td><strong>OracleRowsCopiedEventHandler Delegate</strong></td>
<td>The OracleRowsCopiedEventHandler delegate represents the method that handles the OracleRowsCopied event of an OracleBulkCopy object.</td>
</tr>
<tr>
<td><strong>OracleRowsCopiedEventArgs Class</strong></td>
<td>The OracleRowsCopiedEventArgs class represents the set of arguments passed as part of event data for the OracleRowsCopied event.</td>
</tr>
<tr>
<td><strong>OracleRowUpdatedEventArgs Class</strong></td>
<td>The OracleRowUpdatedEventArgs object provides event data for the OracleDataAdapter.RowUpdated event.</td>
</tr>
<tr>
<td><strong>OracleRowUpdatedEventHandler Delegate</strong></td>
<td>The OracleRowUpdatedEventHandler delegate represents the signature of the method that handles the OracleDataAdapter.RowUpdated event.</td>
</tr>
<tr>
<td><strong>OracleRowUpdatingEventArgs Class</strong></td>
<td>The OracleRowUpdatingEventArgs object provides event data for the OracleDataAdapter.RowUpdating event.</td>
</tr>
<tr>
<td><strong>OracleRowUpdatingEventHandler Delegate</strong></td>
<td>The OracleRowUpdatingEventHandler delegate represents the signature of the method that handles the OracleDataAdapter.RowUpdating event.</td>
</tr>
<tr>
<td><strong>OracleTransaction Class</strong></td>
<td>An OracleTransaction object represents a local transaction.</td>
</tr>
<tr>
<td></td>
<td><em>Not Supported in a .NET Stored Procedure</em></td>
</tr>
<tr>
<td><strong>OracleXmlQueryProperties Class</strong></td>
<td>An OracleXmlQueryProperties object represents the XML properties used by the OracleCommand class when the XmlCommandType property is Query.</td>
</tr>
<tr>
<td><strong>OracleXmlSaveProperties Class</strong></td>
<td>An OracleXmlSaveProperties object represents the XML properties used by the OracleCommand class when the XmlCommandType property is Insert, Update, or Delete.</td>
</tr>
</tbody>
</table>

### Oracle.DataAccess.Client Enumerations

Table 1–2 lists the 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 specify the state of the failover.</td>
</tr>
<tr>
<td></td>
<td><em>Not Supported in a .NET Stored Procedure</em></td>
</tr>
<tr>
<td>FailoverReturnCode Enumeration</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></td>
<td><em>Not Supported in a .NET Stored Procedure</em></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></td>
<td><em>Not Supported in a .NET Stored Procedure</em></td>
</tr>
<tr>
<td>OracleAQDequeueMode Enumeration</td>
<td>The OracleAQDequeueMode enumeration type specifies the dequeue mode.</td>
</tr>
<tr>
<td>OracleAQMessageDeliveryMode Enumeration</td>
<td>The OracleAQMessageDeliveryMode enumeration type specifies the delivery mode of the message.</td>
</tr>
<tr>
<td>OracleAQMessageState Enumeration</td>
<td>The OracleAQMessageState enumeration type identifies the state of the message at the time of dequeue.</td>
</tr>
<tr>
<td>OracleAQMessageType Enumeration</td>
<td>The OracleAQMessageType enumeration type specifies the message payload type.</td>
</tr>
<tr>
<td>OracleAQNavigationMode Enumeration</td>
<td>The OracleAQNavigationMode enumeration type specifies the navigation mode.</td>
</tr>
<tr>
<td>OracleAQNotificationGroupingType Enumeration</td>
<td>The OracleAQNotificationGroupingType enumeration type specifies the notification grouping type.</td>
</tr>
<tr>
<td>OracleAQNotificationType Enumeration</td>
<td>The OracleAQNotificationType enumeration type specifies the notification type of the received notification.</td>
</tr>
<tr>
<td>OracleAQVisibilityMode Enumeration</td>
<td>The OracleAQVisibilityMode enumeration type specifies whether the enqueue or dequeue operation is part of the current transaction.</td>
</tr>
<tr>
<td>OracleBulkCopyOptions Enumeration</td>
<td>The OracleBulkCopyOptions enumeration specifies the values that can be combined with an instance of the OracleBulkCopy class and used as options to determine its behavior and the behavior of the WriteToServer methods for that instance.</td>
</tr>
<tr>
<td>OracleCollectionType Enumeration</td>
<td>OracleCollectionType enumerated values specify whether or not the OracleParameter object represents a collection, and if so, specifies the collection type.</td>
</tr>
<tr>
<td></td>
<td><em>Not Supported in a .NET Stored Procedure</em></td>
</tr>
<tr>
<td>OracleDBShutdownMode Enumeration</td>
<td>OracleDBShutdownMode enumerated values specify the database shutdown options.</td>
</tr>
<tr>
<td>OracleDBStartupMode Enumeration</td>
<td>OracleDBStartupMode enumerated values specify the database startup options.</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 type 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 data type.</td>
</tr>
</tbody>
</table>
Oracle.DataAccess.Types Exceptions

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

<table>
<thead>
<tr>
<th>Exception Class</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 Types classes.</td>
</tr>
<tr>
<td>OracleNullValueException Class</td>
<td>The OracleNullValueException represents an exception that is thrown when trying to access an ODP.NET Types 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 Types class occurs.</td>
</tr>
</tbody>
</table>

Oracle.DataAccess.Types Classes

Table 1–5 lists the type classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleArrayMappingAttribute Class</td>
<td>The OracleArrayMappingAttribute class is required to mark a custom class field or property with information that ODP.NET uses when a custom type represents an Oracle Collection type.</td>
</tr>
<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 BFILE objects.</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 BLOB objects.</td>
</tr>
</tbody>
</table>
**Oracle.DataAccess.Types Interfaces**

Table 1–6 lists the type interfaces.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOracleArrayTypeFactory Interface</td>
<td>The IOracleArrayTypeFactory interface is used by ODP.NET to create arrays that represent Oracle Collections.</td>
</tr>
<tr>
<td>IOracleCustomType Interface</td>
<td>IOracleCustomType is an interface for converting between a Custom Type and an Oracle Object or Collection Type.</td>
</tr>
<tr>
<td>IOracleCustomTypeFactory Interface</td>
<td>The IOracleCustomTypeFactory interface is used by ODP.NET to create custom objects that represent Oracle Objects or Collections.</td>
</tr>
<tr>
<td>INullable Interface</td>
<td>The INullable interface is used to determine whether or not an ODP.NET type has a NULL value.</td>
</tr>
</tbody>
</table>

**Oracle.DataAccess.Types Enumerations**

Table 1–7 lists the type enumerations.

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleUdtFetchOption Enumeration</td>
<td>OracleUdtFetchOption enumeration values specify how to retrieve a copy of the referenceable object.</td>
</tr>
</tbody>
</table>
Using ODP.NET Client Provider in a Simple Application

The following is a simple C# application that connects to 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();
}
}
```

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

### Table 1–7 (Cont.) Oracle.DataAccess.Types Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleUdtStatus Enumeration</td>
<td>OracleUdtStatus enumeration values specify the status of an object attribute or collection element. An object attribute or a collection element can be a valid value or a null value.</td>
</tr>
</tbody>
</table>
This chapter describes installation and configuration requirements for Oracle Data Provider for .NET.

This chapter contains these topics:

- System Requirements
- Oracle Data Provider for .NET Versioning Scheme
- Installing Oracle Data Provider for .NET
- Configuring Oracle Data Provider for .NET
- Configuring a Port to Listen for Database Notifications
- General .NET Programming Recommendations and Tips for ODP.NET

## System Requirements

Oracle Data Provider for .NET requires the following:

- Windows operating system

  Oracle supports 32-bit ODP.NET on x86, AMD64, and Intel EM64T processors on these operating systems.

  - 64-bit: Windows 7 x64 (Professional, Enterprise, and Ultimate Editions), Windows Vista x64 (Business, Enterprise, and Ultimate Editions), Windows Server 2008 R2 x64 (Standard, Enterprise, Datacenter, Web, and Foundation Editions), Windows Server 2008 x64 (Standard, Enterprise, Datacenter, Web, and Foundation Editions), Windows Server 2003 x64 (all editions), Windows Server 2003 R2 x64 (all editions), or Windows XP x64.

  Oracle supports 32-bit ODP.NET and 64-bit ODP.NET for Windows x64 on these operating systems.

---

**Note:** ODP.NET does not support Itanium systems.
Microsoft .NET Framework
- ODP.NET for .NET Framework 2.0 is only supported with Microsoft .NET Framework 2.0, 3.0, or 3.5.
- ODP.NET for .NET Framework 4 is only supported with Microsoft .NET Framework 4 or .NET Framework 4 Client Profile.

Access to Oracle9i Database Release 2 or later

Oracle Client release 11.2

This is automatically installed as part of the ODP.NET installation.

Additional requirements are the following:

Applications using promotable and distributed transactions require Oracle Services for Microsoft Transaction Server 11.2. ODP.NET only supports the read committed isolation level for distributed transactions.

Applications using OracleXmlStream and OracleXmlType classes with schema-based XMLType require access to Oracle Database 10g release 1 (10.1) or later.

See Also:
- Document 726240.1 on My Oracle Support (formerly OracleMetaLink) for details on supported configuration for different ODP.NET versions. You can access My Oracle Support from:
  https://support.oracle.com
- http://msdn.microsoft.com/netframework
- http://otn.oracle.com/tech/xml/xdkhome.html to download the Oracle XML Developer's Kit (XDK)

Oracle Data Provider for .NET Versioning Scheme

Starting with 11.2.0.1.2, Oracle Data Provider for .NET ships with two sets of binaries: one set for .NET Framework 2.0 and another for .NET Framework 4.

For example, ODP.NET 11.2.0.1.2 binaries would be the following:

ODP.NET for .NET Framework 4
- Oracle.DataAccess.dll
  * Built with .NET Framework 4
  * Assembly version number: 4.x.x.x
- OraOps11w.dll
  * Used by ODP.NET for .NET Framework 2.0 and 4

ODP.NET for .NET Framework 2.0
- Oracle.DataAccess.dll
  * Built with .NET Framework 2.0
  * Assembly version number: 2.x.x.x
- OraOps11w.dll
  * Used by ODP.NET for .NET Framework 2.0 and 4
The convention for ODP.NET assembly/DLL versioning is
\[n1.o1o2.o3o4.o5\]
where:
- \(n1\) is the most significant .NET Framework version number.
- \(o1o2\) are the first two digits of the ODP.NET product version number.
- \(o3o4\) are the third and forth digits of the ODP.NET product version number.
- \(o5\) is the fifth and last digit of the ODP.NET product version number.

For example, if the ODP.NET product version number is 11.2.0.2, the corresponding ODP.NET assembly versions are:
- .NET Framework 4 version: 4.112.2.0
- .NET Framework 2.0 version: 2.112.2.0

Note that the Oracle installer and documentation still refer to the ODP.NET product version number and not the assembly/DLL version number.

As with the .NET Framework system libraries, the first digit of the assembly version number indicates the version of the .NET Framework to use with an ODP.NET assembly.

Publisher Policy DLL is provided as before so that applications built with older versions of ODP.NET are redirected to the newer ODP.NET assembly, even though the versioning scheme has changed.

### Installing Oracle Data Provider for .NET

Oracle Data Provider for .NET is part of Oracle Data Access Components (ODAC), which can be downloaded from OTN. Beginning with ODAC 11.1.0.6.20, Oracle Data Provider for .NET can be installed through XCopy or Oracle Universal Installer.

- **XCopy**
  
  Administrators use XCopy to deploy Oracle Data Provider for .NET to large numbers of computers for production deployments. The XCopy has a smaller installation size and fine-grain control during installation and configuration than Oracle Universal Installer.

- **Oracle Universal Installer (OUI)**
  
  Developers and administrators use Oracle Universal Installer for automated ODP.NET installations. It includes documentation and code samples that are not part of the XCopy.

---

**Note:** This section describes installation using the Oracle Universal Installer. For installation and configuration using the XCopy install, refer to the README.TXT file that is part of the XCopy installation.

---

Additionally, Oracle Data Provider for .NET Dynamic Help is registered with Visual Studio, providing context-sensitive online help that is seamlessly integrated with Visual Studio Dynamic Help. With Dynamic Help, the user can access ODP.NET documentation within the Visual Studio IDE by placing the cursor on an ODP.NET keyword and pressing the F1 function key.
Oracle Data Provider for .NET creates an entry in the `machine.config` file of the computer on which it is installed, for applications using ADO.NET 2.0 and `OracleClientFactory` class. This enables the `DbProviderFactories` class to recognize ODP.NET.

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

### File Locations After Installation

The `Oracle.DataAccess.dll` assembly is installed to the following locations:

- **.NET Framework 2.0:**
  - `ORACLE_BASE\ORACLE_HOME\odp.net\bin\2.x` directory
- **.NET Framework 4:**
  - `ORACLE_BASE\ORACLE_HOME\odp.net\bin\4` directory

**Note:** If the machine has the corresponding .NET Framework installed, then the `Oracle.DataAccess.dll` assembly is added to the Global Assembly Cache (GAC) as well. This is to ensure that existing applications can start using the newly installed ODP.NET version immediately. However, if this is not desired, be sure to remove the policy DLLs from the GAC.

Documentation and the `readme.txt` file 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.

### Search Order for Unmanaged DLLs

ODP.NET consists of managed and unmanaged binaries. Through the use of the `DllPath` configuration parameter, each application can specify the `ORACLE_BASE\ORACLE_HOME\bin` location that the dependent unmanaged Oracle Client binaries are loaded from. However, the `ORACLE_BASE\ORACLE_HOME` must have the same ODP.NET version installed as the version that the application uses. Otherwise, a version mismatch exception is thrown.

The `Oracle.DataAccess.dll` searches for dependent unmanaged DLLs (such as Oracle Client) based on the following order:

1. Directory of the application or executable.
2. `DllPath` setting specified by application `config` or `web.config`.
3. `DllPath` setting specified by `machine.config`.
4. `DllPath` setting specified by the Windows Registry.
   - `HKEY_LOCAL_MACHINE\Software\Oracle\ODP.NET\version\DllPath`.
5. Directories specified by the Windows `PATH` environment variable.

Upon installation of ODP.NET, Oracle Universal Installer sets the `DllPath` Windows Registry value to the `ORACLE_BASE\ORACLE_HOME\bin` directory where the
corresponding dependent DLLs are installed. Developers must provide this configuration information on an application-by-application basis.

When a new ODP.NET version is installed, default values are set in the Windows Registry for the new version. Because the policy DLLs redirect all ODP.NET references to this new ODP.NET version, applications use the default values. Developers can provide a config or web.config file specific to the application to prevent this redirection. The configuration file settings always apply to the application, regardless of whether or not patches or new versions are installed later.

ODP.NET Configuration File Support is only available for version 10.2.0.4 and later.

---

**Note:** Both Oracle.DataAccess.dll for .NET Framework 2.0 and Oracle.DataAccess.dll for .NET Framework 4 use the same unmanaged DLL, OraOps11w.dll.

---

**ODP.NET and Dependent Unmanaged DLL Mismatch**

To enforce the usage of Oracle.DataAccess.dll assembly with the correct version of its unmanaged DLLs, an exception is raised if Oracle.DataAccess.dll notices it has loaded a mismatched version of a dependent unmanaged DLL.

---

**Configuring Oracle Data Provider for .NET**

The settings for specific versions of ODP.NET can be configured in several ways for specific effects on precedence:

- The Windows registry entries are machine-wide settings for a particular version of ODP.NET.
- The machine.config settings are .NET framework-wide settings that override the Windows registry values.
- The application or web config file settings are application-specific settings that override the machine.config settings and the Windows registry settings.
- Any attribute settings that are equivalent to the connection string override everything.

The application or web config file can be useful and sometimes essential in scenarios where more than one application on a computer use the same version of ODP.NET, but each application needs a different ODP.NET configuration. The Windows registry value settings for a given version of ODP.NET affect all the applications that use that version of ODP.NET. However, having ODP.NET configuration values in the application or web config file assure that these settings are applied only for that application, thus providing more granularity.

For example, if the application or web.config file has a StatementCacheSize configuration setting of 100, this application-specific setting forces the version of ODP.NET that is loaded by that application to use 100 for the StatementCacheSize and overrides any setting in the machine.config and in the registry. Note that for any setting that does not exist in a config file (machine.config or application/web config), the value in the registry for a loaded version of ODP.NET is used, as in previous releases.

Note that ODP.NET reads the machine.config files from the version of the .NET Framework on which ODP.NET runs, not from the version of ODP.NET.
ODP.NET only reads the Windows Registry and the XML configuration file when it is loaded into memory, thus any configuration changes made after that are not read or used until the application is re-started.

**Supported Configuration Settings**

ODP.NET supports the configuration of an attribute as follows:

- In the Windows registry.
- In an XML file.
- Through a different mechanism such as a connection string or programmatically through an ODP.NET class, if applicable.

**Table 2–1** describes each configurable attribute that is supported by ODP.NET. In the table, the term Configuration Support is followed by the types of configuration support (Windows registry, XML file, and so on) that are available for that attribute. The table describes valid values as well as the default for each attribute.

---

**Note:** The default values shown are the values used for an attribute if the registry key does not exist or if it is not configured anywhere.

---

**Table 2–1  Configuration Attributes**

<table>
<thead>
<tr>
<th>Attribute/Setting Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckConStatus</td>
<td>Specifies whether the status of the connection is checked or not before putting the connection back into the connection pool. This registry entry is not created by the installation of ODP.NET. However, the default value 1 is used.</td>
</tr>
<tr>
<td></td>
<td>Configuration Support:</td>
</tr>
<tr>
<td></td>
<td>Windows Registry and XML file</td>
</tr>
<tr>
<td>Valid Values:</td>
<td></td>
</tr>
<tr>
<td>1: Check the status of the connection.</td>
<td></td>
</tr>
<tr>
<td>0: Do not check the status of the connection.</td>
<td></td>
</tr>
<tr>
<td>Default: 1</td>
<td></td>
</tr>
<tr>
<td>DbNotificationPort</td>
<td>Specifies the port number which ODP.NET listens to, for all notifications sent by the database for change notification, HA, or RLB features. ODP.NET does not throw any errors if an invalid or used port number is specified. The port can also be set to override the Windows registry and XML configuration file by setting the <code>OracleDependency.Port</code> static field.</td>
</tr>
<tr>
<td></td>
<td>Configuration Support:</td>
</tr>
<tr>
<td></td>
<td>XML file, and ODP.NET class</td>
</tr>
<tr>
<td>Valid Values:</td>
<td></td>
</tr>
<tr>
<td>-1: Open a random unused port to listen to.</td>
<td></td>
</tr>
<tr>
<td>( n &gt; 0 ): Listen on port ( n ).</td>
<td></td>
</tr>
<tr>
<td>Default: -1</td>
<td></td>
</tr>
</tbody>
</table>
### Table 2–1 (Cont.) Configuration Attributes

<table>
<thead>
<tr>
<th>Attribute/Setting Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DemandOraclePermission</td>
<td>Specifies whether ODP.NET demands OraclePermission from the .NET application that is trying to access the database using ODP.NET. Configuration Support: Windows Registry and XML file. Valid Values: 0: Disables demands for OraclePermission. 1: Enables demands for OraclePermission. Default: 0.</td>
</tr>
<tr>
<td>DllPath</td>
<td>Specifies the location where dependent unmanaged Oracle Client binaries load from. Configuration Support: Windows Registry and XML file. Valid Values: The path where dependent unmanaged Oracle Client binaries reside. Default: <code>ORACLE_BASE\ORACLE_HOME\bin</code>.</td>
</tr>
<tr>
<td>DynamicEnlistment</td>
<td>Specifies whether the application enlists in distributed transactions explicitly after an OracleConnection.Open method invocation through EnlistTransaction() or EnlistDistributedTransaction(). To configure ODP.NET to enable dynamic enlistment programmatically, the connection string must contain &quot;enlist=dynamic&quot;. Configuration Support: Windows Registry, XML file, and &quot;enlist&quot; connection string attribute. Valid Values: 0: Disables ability to explicitly enlist in distributed transactions. 1: Enables ability to explicitly enlist in distributed transactions. Default: 0.</td>
</tr>
<tr>
<td>FetchSize</td>
<td>Specifies the total memory size, in bytes, that ODP.NET allocates to cache the data fetched from a database round-trip. This value can be set on the OracleCommand and the OracleDataReader FetchSize property as well. Configuration Support: Windows Registry, XML file, and ODP.NET class. Valid Values: ( 0 \leq n \leq \text{int.MaxValue} ): ( n ) is the size of the cache in bytes. Default: 131072.</td>
</tr>
</tbody>
</table>
Table 2–1  (Cont.) Configuration Attributes

<table>
<thead>
<tr>
<th>Attribute/Setting Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaDataXml</td>
<td>Specifies the name of the XML file that customizes the queries to obtain the metadata the ADO.NET 2.0 GetSchema method returns. MetaDataXml can only be set in a configuration file.</td>
</tr>
</tbody>
</table>
|                        | Configuration Support:  
|                        | XML file only  
|                        | Valid Values:  
|                        | A complete file name for the XML file.  
|                        | Default: none |
| PerformanceCounters    | Enables or disables publishing performance counters for connection pooling. Multiple performance counters can be obtained by adding the valid values. |
|                        | Configuration Support:  
|                        | Windows Registry and XML file  
|                        | Valid Values:  
|                        | 0: Not Enabled  
|                        | 1: Number of sessions being established with Oracle Database every second.  
|                        | 2: Number of sessions being severed from Oracle Database every second.  
|                        | 4: Number of active connections originating from connection pools every second.  
|                        | 8: Number of active connections going back to the connection pool every second.  
|                        | 16: Total number of active connections.  
|                        | 32: Number of inactive connection pools.  
|                        | 64: Total number of connections in use.  
|                        | 128: Total number of connections available for use in all the connection pools.  
|                        | 256: Number of pooled active connections.  
|                        | 1024: Number of non-pooled active connections.  
|                        | 2048: Number of connections that will be soon available in the pool. User has closed these connections, but they are currently awaiting actions, such transaction completion, before they can be placed back into the pool as free connections.  
|                        | 4095: All the above  
|                        | Default: 0 |
PromotableTransaction
Specifies the type of transaction to use when the first connection participates in the TransactionScope object.

Configuration Support:
Windows Registry, XML file, and promotable transaction connection string attribute

Valid Values:
local: The first connection opened in the TransactionScope object uses a local transaction.
promotable: The first connection and all subsequent connections opened in the same TransactionScope object enlist in the same distributed transaction.

Default: promotable

SelfTuning
Specifies whether self-tuning is enabled for an ODP.NET application.

Configuration Support:
Windows Registry, XML file, and Self Tuning connection string attribute

Valid Values:
0: Self Tuning is disabled. Used in the registry or XML file.
false: Self Tuning is disabled. Used for the Self Tuning connection string attribute.
1: Self Tuning is enabled. Used in the registry or XML file.
true: Self Tuning is enabled. Used for the Self Tuning connection string attribute.

Default: 1

StatementCacheSize
Specifies the number of cursors or statements to be cached on the database for each connection. This setting corresponds to Statement Cache Size attribute in the connection string. A value greater than zero also enables statement caching.

Configuration Support:
Windows Registry, XML file, and Statement Cache Size connection string attribute

Valid Values:
0 <= n <= the value of OPEN_CURSORS parameter set in init.ora database config file.
n is the number to set.

Default: 0

StatementCacheWithUdts
Specifies whether or not Oracle UDTs retrieved by executing a SELECT statement are cached along with the statement in the statement cache. This setting affects the memory usage and performance of the application.

Configuration Support:
Windows Registry and XML file

Valid Values:
0: Oracle UDTs are not cached with statements.
1: Oracle UDTs are cached along with statements.

Default: 1
ThreadPoolMaxSize
Specifies the default maximum size of worker threads for each available processor in a process. This value may affect the performance of ODP.NET connection creation, command execution timeout, and external procedures (extproc) that use the thread pool. However, unnecessarily increasing thread pool maximum size can also cause performance problems.

Configuration Support:
Windows Registry and XML file
Valid Values:
0 <= n <= int.MaxValue: Allows ODP.NET to reset thread pool maximum size with the value n. The ODP.NET reset operation may be ignored if the value is invalid. For example, if n is less than the number of available processors of the system. In this case, the result is the same as the value -1.

-1: Leave the thread pool max size as is.
Default: -1 (this registry entry is not created by default)

Note that prior to ODAC 2007 or version 11.1.0.6.20, ODP.NET resets the thread pool maximum size to int.MaxValue when the OracleCommand.CommandTimeout property is set to a value greater than 0. This erroneous behavior has been corrected. OracleCommand.CommandTimeout does not change thread pool maximum size.

TraceFileName
Specifies the file name to be used for logging trace information.

Configuration Support:
Windows Registry and XML file
Valid Values:
Any valid directory location and file name.
Default: c:\odpnet2.trc (for .NET Framework 2.0)
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. Each valid value indicates a possible tracing level. Compounded tracing levels can be obtained by adding the valid values.

Configuration Support:
Windows Registry and XML file
Valid Values:
0: None
1: Entry, exit, and SQL statement information
2: Connection pooling statistics
4: Distributed transactions (enlistment and delistment)
8: User-mode dump creation upon unmanaged exception
16: HA Event Information
32: Load Balancing Information
64: Self Tuning Information
127: All the above
Default: 0

Note: ODP.NET does bit-wise checking on the value. When tracing is enabled, logging to the trace file can affect ODP.NET performance.

Note: The user-mode dump creation requires dbghelp.dll version 5.1.2600.0 or later.

TraceOption

Specifies whether to log trace information in single or multiple files for different threads. If a single trace file is specified, the file name specified in TraceFileName is used. If the multiple trace files option is requested, a Thread ID is appended to the file name provided to create a trace file for each thread.

Configuration Support:
Windows Registry and XML file
Valid Values:
0: Single trace file
1: Multiple trace files
Default: 0

UdtCacheSize

Specifies the size of the object cache for each connection in kilobytes (KB) that ODP.NET uses to retrieve and manipulate Oracle UDTs.

Configuration Support:
Windows Registry and XML file
Valid Values:
0 <= n <= 4194303, n is the number to set.
Default: 4096
Upon installation, ODP.NET creates entries for configuration and tracing within the Windows Registry. Configuration and tracing registry values apply across all ODP.NET applications running in that Oracle client installation. Individual ODP.NET applications can override some of these values by configuring them within the ODP.NET application itself (for example, `FetchSize`). Applications can also use the .NET configuration files to override some of the ODP.NET Windows Registry values.

The ODP.NET registry values are located under `HKEY_LOCAL_MACHINE\Software\Oracle\ODP.NET\version\`. There is one key for .NET Framework 2.0, 3.0, and 3.5, and one key for .NET Framework 4 and higher.

---

**Note:** 32-bit applications running on an x64-based version of Windows use the registry subkey, `HKEY_LOCAL_MACHINE\Software\Wow6432Node` in place of `HKEY_LOCAL_MACHINE\Software`. If such applications use Oracle Data Provider for .NET (32-bit), then the ODP.NET registry values are located under `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Oracle\ODP.NET\version\`.

---

### Configuration File Support

For customers who have numerous applications on a computer that depends on a single version of ODP.NET, the Windows Registry settings for a given version of ODP.NET may not necessarily be applicable for all applications that use that version of ODP.NET. To provide more granular control, ODP.NET Configuration File Support allows developers to specify ODP.NET configuration settings in an application config, `web.config`, or a `machine.config` file.

If a computer does not require granular control beyond configuration settings at the ODP.NET version level, there is no need to specify ODP.NET configuration settings through configuration files.

The following is an example of a `web.config` file for .NET Framework 2.0 and higher:

```xml
<configuration>
</configuration>
```
Configuring a Port to Listen for Database Notifications

Oracle Data Provider for .NET opens a port to listen for database notifications when the following features are used:

- HA Events
- Load Balancing
- Database change notification
- AQ Notifications

All these features share the same port, which can be configured centrally by setting the db notifications port in an application or web configuration file.

Sample Configuration Files

The following is an example of app.config for ODP.NET using .NET Framework 2.0, which sets some additional attributes as well as two UDT type mappings:

```xml
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DbNotificationPort" value="-1"/>
      <add name="DllPath" value="C:\app\user\product\11.1.0\client_1\bin"/>
      <add name="DynamicEnlistment" value="0"/>
      <add name="FetchSize" value="131072"/>
      <add name="MetaDataXml" value="CustomMetaData.xml"/>
      <add name="PerformanceCounters" value="4095"/>
      <add name="PromotableTransaction" value="promotable"/>
      <add name="ThreadPoolMaxSize" value="30"/>
      <add name="StatementCacheSize" value="50"/>
      <add name="TraceFileName" value="c:\odpnet2.trc"/>
      <add name="TraceLevel" value="0"/>
      <add name="TraceOption" value="0"/>
      <add name="Person" value="udtMapping factoryName='PersonFactory, Sample, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' typeName='PERSON' schemaName='SCOTT' dataSource='oracle'"/>
      <add name="Student" value="udtMapping factoryName='StudentFactory, Sample, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' typeName='STUDENT' schemaName='SCOTT'"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>

If the configuration file does not exist or the db notification port is not specified, ODP.NET uses a valid, random port number. The configuration file may also request for a random port by specifying a db notification port value of -1. To specify a particular port, for example, 1200, an application or web configuration file can be used as shown below.

```xml
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DbNotificationPort" value="1200"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

The port number should be unique for each process running on a computer. Thus, the port number should be set uniquely for each application either programmatically or through an application config file. Note that if the specified port number is already in use or invalid, ODP.NET does not provide any errors.

When the process using ODP.NET starts, the application reads the db notification port number and listens on that port. Once the port is opened, the port number cannot be changed during the lifetime of the process.

See Also:
- "HA Events" on page 3-11
- "Runtime Connection Load Balancing" on page 3-10
- "Database Change Notification Support" on page 3-120
- "Oracle Streams Advanced Queuing Support" on page 3-115

General .NET Programming Recommendations and Tips for ODP.NET

- Thread.Abort() should not be used, as unmanaged resources may remain unreleased, which can potentially cause memory leaks and hangs.

- To optimize resource usage, ODP.NET objects, such as OracleConnection and OracleCommand, should be explicitly closed or disposed, or both, when they are no longer needed. This should be done rather than relying on the .NET Framework garbage collector to reclaim resources. Many users have found that under stress conditions, explicit Close or Dispose calls result in much lower resource usage.

- It is recommended not to proceed with application execution if the application encounters exceptions that are associated with possible memory corruption, such as System.AccessViolationException and System.Runtime.InteropServices.SEHException.

- If the HKEY_LOCAL_MACHINE\Software\Oracle\NLS_LANG registry entry is set to NA, ODP.NET encounters ORA-12705 errors. To eliminate this problem, remove the HKEY_LOCAL_MACHINE\Software\Oracle\NLS_LANG registry entry.
Features of Oracle Data Provider for .NET

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

This chapter contains these topics:

- Connecting to Oracle Database
- ADO.NET 2.0 Features
- ADO.NET Entity Framework and LINQ to Entities
- OracleCommand Object
- ODP.NET Types Overview
- Obtaining Data from an OracleDataReader Object
- PL/SQL REF CURSOR and OracleRefCursor
- Implicit REF CURSOR Binding
- LOB Support
- ODP.NET XML Support
- Oracle User-Defined Types (UDTs) and .NET Custom Types
- Oracle Streams Advanced Queuing Support
- Database Change Notification Support
- OracleDataAdapter Safe Type Mapping
- OracleDataAdapter Requery Property
- Guaranteeing Uniqueness in Updating DataSet to Database
- Globalization Support
- Debug Tracing

Connecting to Oracle Database

Oracle Data Provider for .NET can connect to Oracle Database in a number of ways, such as using a username and password, Windows Native Authentication, Kerberos, and Secure Sockets Layer (SSL). This section describes OracleConnection provider-specific features, including:

- Connection String Attributes
- Connection Pooling
Connection String Attributes

Table 3-1 lists the supported connection string attributes.

<table>
<thead>
<tr>
<th>Connection String Attribute</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Lifetime</td>
<td>Maximum life time (in seconds) of the connection.</td>
<td>0</td>
</tr>
<tr>
<td>Connection Timeout</td>
<td>Maximum time (in seconds) to wait for a free connection from the pool.</td>
<td>15</td>
</tr>
<tr>
<td>Context Connection</td>
<td>Returns an implicit database connection if set to true. <em>Supported in a .NET stored procedure only</em></td>
<td>false</td>
</tr>
<tr>
<td>Data Source</td>
<td>Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.</td>
<td>empty string</td>
</tr>
<tr>
<td>DBA Privilege</td>
<td>Administrative privileges: SYSDBA or SYSOPER.</td>
<td>empty string</td>
</tr>
<tr>
<td>Decr Pool Size</td>
<td>Number of connections that are closed when an excessive amount of established connections are unused.</td>
<td>1</td>
</tr>
<tr>
<td>Enlist</td>
<td>Controls the enlistment behavior and capabilities of a connection in context of COM+ transactions or System.Transactions.</td>
<td>true</td>
</tr>
<tr>
<td>HA Events</td>
<td>Enables ODP.NET connection pool to proactively remove connections from the pool when an Oracle RAC service, service member, database, or node goes down. Works with Oracle RAC, Data Guard, or a single database instance.</td>
<td>false</td>
</tr>
</tbody>
</table>
The following example uses connection string attributes to connect to 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 5-76 for detailed information on connection attributes
- "OracleCommand Object" on page 3-44 for detailed information on statement caching

Specifying the Data Source Attribute

This section describes different ways of specifying the data source attribute.

The following example shows a connect descriptor mapped to a TNS alias called sales in the tnsnames.ora file:

sales=
  (DESCRIPTION=
    (ADDRESS= (PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
    (CONNECT_DATA=
      (SERVICE_NAME=sales.us.acme.com)))

Using the TNS Alias

To connect as scott/tiger using the TNS Alias, a valid connection appears as follows:

"user id=scott;password=tiger;data source=sales";

Using the Connect Descriptor

ODP.NET also allows applications to connect without the use of the tnsnames.ora file. To do so, the entire connect descriptor can be used as the "data source".

The connection string appears as follows:

"user id=scott;password=tiger;data source=" +
  "(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)" +
  "(HOST=sales-server)(PORT=1521))" +
  "(CONNECT_DATA=" +
  "(SERVICE_NAME=sales.us.acme.com))")"

Using Easy Connect Naming Method

The easy connect naming method enables clients to connect to a database without any configuration.

Prior to using the easy connect naming method, make sure that EZCONNECT is specified by the NAMESDIRECTORY_PATH parameter in the sqlnet.ora file as follows:

NAMESDIRECTORY_PATH= (TNSNAMES, EZCONNECT)
With this enabled, ODP.NET allows applications to specify the "Data Source" attribute in the form of:

```java
//host:[port]/[service_name]
```

Using the same example, some valid connection strings follow:

- "user id=scott;password=tiger;data source=//sales-server:1521/sales.us.acme.com"
- "user id=scott;password=tiger;data source=//sales-server/sales.us.acme.com"
- "user id=scott;password=tiger;data source=sales-server/sales.us.acme.com"

If the port number is not specified, 1521 is used by default.

See Also: Oracle Net Services Administrator’s Guide for details and requirements in the section Using Easy Connect Naming Method

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:

- Connection Lifetime
- Connection Timeout
- Decr Pool Size
- HA Events
- Incr Pool Size
- Load Balancing
- Max Pool Size
- Min Pool Size
- Pooling
- Validate Connection

Connection Pooling Example

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

```csharp
// 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;" +

       con.Open();       Console.WriteLine("Connected to Oracle Database.");
   }
}
```
"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**

When connection pooling is enabled (the default), the Open and Close methods of the OracleConnection object implicitly use 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 property as a signature, to uniquely identify a pool.

If there is no existing pool with the exact attribute values as the ConnectionString property, 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 property. This number of connections is always maintained by the connection pooling service for the connection pool.

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

The **Incr Pool Size** attribute of the ConnectionString property 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 or not the connection lifetime has exceeded the value of 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 enforces the Connection Lifetime only when a connection is going back to the connection pool.

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

The **Validate Connection** attribute validates connections coming out of the pool. This attribute should be used only when absolutely necessary, because it causes a round-trip to the database to validate each connection immediately before it is provided to the application. If invalid connections are uncommon, developers can create their own event handler to retrieve and validate a new connection, rather than using the **Validate Connection** attribute. This generally provides better performance.
The connection pooling service closes connections when they are not used; connections are closed every 3 minutes. The `Decr Pool Size` attribute of the `ConnectionString` property provides connection pooling service for the maximum number of connections that can be closed every 3 minutes.

Beginning with Oracle Data Provider for .NET release 11.1.0.6.20, enabling connection pooling by setting "pooling=true" in the connection string (which is the case by default) will also pool operating system authenticated connections.

### Connection Pool Management

ODP.NET connection pool management provides explicit connection pool control to ODP.NET applications. Applications can explicitly clear connections in a connection pool.

Using connection pool management, applications can do the following:

- Clear connections from connection pools using the `ClearPool` method.
- Clear connections in all the connection pools in an application domain, using the `ClearAllPools` method.

When connections are cleared from a pool, ODP.NET repopulates the pool with new connections that have at least the number of connections set by `Min Pool Size` in the connection string. New connections do not necessarily mean the pool will have valid connections. For example, if the database server is down when `ClearPool` or `ClearAllPools` is called, ODP.NET creates new connections, but these connections are still invalid because they cannot connect to the database, even if the database comes up a later time.

It is recommended that `ClearPool` and `ClearAllPools` not be called until the application can create valid connections back to the database. .NET developers can develop code that continuously checks whether or not a valid database connection can be created and calls `ClearPool` or `ClearAllPools` once this is true.

**See Also:**
- "ClearPool" on page 5-73
- "ClearAllPools" on page 5-74

### Connection Pool Performance Counters

Installing Oracle Data Provider for .NET creates a set of performance counters on the target system. These performance counters are published by ODP.NET for each ODP.NET client application. These performance counters can be viewed using Windows Performance Monitor (Perfmon).

In Perfmon, administrators can add ODP.NET counters to the performance monitor graph. ODP.NET performance counters are published under the following Category Name: Oracle Data Provider for .NET. Administrators can choose the ODP.NET counters to monitor after selecting the Oracle Data Provider for .NET category.

As ODP.NET performance counters are not enabled by default, administrators must enable the specific counters of interest before attempting to monitor them. In addition, at least one ODP.NET instance must be actively running when attempting to monitor using Perfmon.

---

**Note:** These APIs are not supported in a .NET stored procedure.
Oracle Data Provider for .NET enables or disables publishing performance counters for connection pooling, using registry entries.

Table 3–2 lists the performance counters used for connection pooling with their valid registry values.

### Table 3–2  Performance Counters for Connection Pooling

<table>
<thead>
<tr>
<th>Performance Counter</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
<td>Not enabled (Default)</td>
</tr>
<tr>
<td>HardConnectsPerSecond</td>
<td>1</td>
<td>Number of sessions being established with the Oracle Database every second.</td>
</tr>
<tr>
<td>HardDisconnectsPerSecond</td>
<td>2</td>
<td>Number of sessions being severed from the Oracle Database every second.</td>
</tr>
<tr>
<td>SoftConnectsPerSecond</td>
<td>4</td>
<td>Number of active connections originating from connection pools every second.</td>
</tr>
<tr>
<td>SoftDisconnectsPerSecond</td>
<td>8</td>
<td>Number of active connections going back to the connection pool every second.</td>
</tr>
<tr>
<td>NumberOfActiveConnectionPools</td>
<td>16</td>
<td>Total number of active connection pools.</td>
</tr>
<tr>
<td>NumberOfInactiveConnectionPools</td>
<td>32</td>
<td>Number of inactive connection pools.</td>
</tr>
<tr>
<td>NumberOfActiveConnections</td>
<td>64</td>
<td>Total number of connections in use.</td>
</tr>
<tr>
<td>NumberOfFreeConnections</td>
<td>128</td>
<td>Total number of connections available for use in all the connection pools.</td>
</tr>
<tr>
<td>NumberOfPooledConnections</td>
<td>256</td>
<td>Number of pooled active connections.</td>
</tr>
<tr>
<td>NumberOfNonPooledConnections</td>
<td>512</td>
<td>Number of non-pooled active connections.</td>
</tr>
<tr>
<td>NumberOfReclaimedConnections</td>
<td>1024</td>
<td>Number of connections which were garbage-collected implicitly.</td>
</tr>
<tr>
<td>NumberOfStasisConnections</td>
<td>2048</td>
<td>Number of connections that will be soon available in the pool. User has closed these connections, but they are currently awaiting actions such transaction completion before they can be placed back into the pool as free connections.</td>
</tr>
</tbody>
</table>

**Publishing Performance Counters**

Publication of individual performance counters is enabled or disabled using the registry value `PerformanceCounters` of type `REG_SZ`. This registry value is under:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version
```

where `Assembly_Version` is the full assembly version number of `Oracle.DataAccess.dll`.

Multiple performance counters can be obtained by adding the valid values. For example, if `PerformanceCounters` is set to 3, both `HardConnectsPerSecond` and `HardDisconnectsPerSecond` are enabled.
Setting Performance Counters Using app.config Entry
Performance counters can be set using an app.config entry. Since app.config entries take precedence over the registry value setting, they can be used for a specific application.

An app.config entry uses name/value pairs as in the following example:

```xml
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="PerformanceCounters" value="3"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

Instance Names of Performance Counters
Performance counters are published instance-wise, that is, for each process, different values of the performance counters are published. The instance name is based on AppDomain name, AppDomain Id and Process Id and displayed in the following form:

AppDomain_Name[Process_Id, AppDomain_Id]

For example, if a process named App1.exe uses ODP.NET 2.x in default appdomain and the process id is 234 then the instance name would be App1.exe[234,1].

See Also: "Connection Pool Performance Counters" on page 3-7

Edition-Based Redefinition
Edition-based redefinition enables you to upgrade the database component of an application even while the application is being used. This minimizes or eliminates downtime for the application.

See Also: For more information on Editions refer to the Oracle Database Administrator’s Guide and Oracle Database Advanced Application Developer’s Guide

ODP.NET 11g Release 2 (11.2.0.1), and higher, supports specifying an Edition at deployment time when used with Oracle Database 11.2 or later. Applications can specify an Edition at deployment time using the registry or configuration file.

An application can create the following registry entry of type REG_SZ:

```
HKLM\Software\Oracle\ODP.NET\version\Edition
```

Here version is the version of ODP.NET, and Edition is a valid Edition string value.

An application can alternatively use the web.config or application.config configuration file to specify the Edition at deployment time. The machine.config configuration file can be used to specify the Edition for all applications that use a particular version of the .NET framework.

The following example sets the Edition to E1 in a configuration file for .NET 2.0, 3.0, and 3.5:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
```
Connecting to Oracle Database

```xml
<oracle.dataaccess.client>
<settings>
  <add name="Edition" value='E1'/>
</settings>
</oracle.dataaccess.client>
</configuration>
```

**Note:** ODP.NET only supports deployment-time configuration of Edition.

ODP.NET does not support usage of the "ALTER SESSION" statement to modify the Edition during the lifetime of a process.

## Connecting to an Oracle Real Application Clusters (Oracle RAC) Database

This section discusses optimization and other aspects of connection and connection pooling for an Oracle Real Application Clusters (Oracle RAC) database. Oracle RAC is the technology that makes grids possible for Oracle database by providing the ability to access the database from multiple instances, each running on nodes in a cluster.

### Connecting in Oracle Real Application Clusters (Oracle RAC) and Data Guard Database

This section discusses optimization and other aspects of connection and connection pooling for Oracle Real Application Clusters (Oracle RAC) and Data Guard databases. Oracle RAC is the technology that makes grids possible for Oracle database by providing the ability to access the database from multiple instances, each running on nodes in a cluster. Oracle Data Guard is a technology that enables high availability and disaster recovery by maintaining a secondary standby database in case the primary database fails.

**See Also:** Oracle Data Guard Concepts and Administration

## Connection Optimization for Oracle RAC and Oracle Data Guard

Oracle Data Provider for .NET supports Oracle RAC and Data Guard transparently. Additionally, the Oracle Data Provider for .NET connection pooling features work with Oracle RAC or Data Guard version 10.2 or higher.

Oracle Data Provider for .NET optimizes connection and connection pooling for Oracle RAC database by balancing work requests across Oracle RAC instances, based on the load balancing advisory and service goal. Furthermore, the ODP.NET connection pool can be enabled to proactively free resources associated with connections that have been severed due to a down Oracle RAC service, service member, node, or database in the case of Data Guard.

Oracle Data Provider for .NET uses the following features to optimize connection and connection pooling for Oracle RAC:

- **Runtime Connection Load Balancing**
  - When Runtime Connection Load Balancing is enabled:
    - The ODP.NET connection pool dispenses connections based on the load balancing advisory and service goal.
    - The ODP.NET connection pool also balances the number of connections to each service member providing the service, based on the load balancing advisory and service goal.
By default, this feature is disabled. To enable runtime connection load balancing, include "Load Balancing=true" in the connection string.

This feature can only be used with an Oracle RAC database and only if "pooling=true". If "Load Balancing=true" is set and the connection attempts to connect to a single-instance database, an OracleException is thrown with an error of "ORA-1031: insufficient privileges."

In order to use Runtime Connection Load Balancing, specific Oracle RAC configurations must be set. For further information, see Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide. Oracle Net Services should also be configured for load balancing. See Oracle Net Services Administrator’s Guide for further details.

The following connection string example enables Runtime Connection Load Balancing:

"user id=scott;password=tiger;data source=erp;load balancing=true;"

See Also:
- "Supported Connection String Attributes" on page 5-79
- "Configuring a Port to Listen for Database Notifications" on page 2-13
- Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide

HA Events

When HA (High Availability) events is enabled, Oracle RAC, Data Guard, and single database instances exhibit the following behavior:

- ODP.NET connection pool proactively removes connections from the pool when an Oracle RAC service, service member, node, or database goes down.
- ODP.NET establishes connections to existing Oracle instances if the removal of severed connections bring the total number of connections below the "min pool size".

By default this feature is disabled. To enable HA events, include "HA Events=true" and "pooling=true" in the connection string.

Note:

The database service being connected to must be configured for AQ\_HA\_NOTIFICATIONS. For more details, see Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide

The following connection string example enables HA Events:

"user id=scott;password=tiger;data source=erp; HA events=true;"
Pool Size Attributes in an Oracle RAC Database

When connection pools are created for a single-instance database, pool size attributes are applied to the single service. Similarly, when connection pools are created for an Oracle RAC database, the pool size attributes are applied to a service and not to service members. For example, if "Min Pool Size" is set to $N$, ODP.NET does not create $N$ connections for each service member. Instead, it creates, at minimum, $N$ connections for the entire service, where $N$ connections are distributed among the service members.

The following pool size connection string attributes are applied to a service.

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

Operating System Authentication

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

\[ \text{Note: } \text{Operating System Authentication is not supported in a .NET stored procedure.} \]

Beginning with Oracle Data Provider for .NET release 11.1.0.6.20, all connections, including those using operating system authentication, can be pooled. Connections are pooled by default, and no configuration is required, as long as pooling is enabled.

The following example shows the use of operating system authentication:

```csharp
/* Create an OS-authenticated user in the database
   Assume init.ora has OS_AUTHENT_PREFIX set to "" and <OS_USER>
   is any valid OS or DOMAIN user.

   create user <OS_USER> identified externally;
   grant connect, resource to <OS_USER>;

   Login through OS Authentication and execute the sample. See Oracle documentation for details on how to configure an OS-Authenticated user */
```

---

See Also:
- "Supported Connection String Attributes" on page 5-79
- "Configuring a Port to Listen for Database Notifications" on page 2-13
- Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide
- Oracle Data Guard Concepts and Administration
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 Oracle Database to authenticate database users using Windows user login credentials

Privileged Connections

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

The following example connects scott/tiger as SYSDBA:

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

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

        //Connect scott/tiger as SYSDBA
        con.ConnectionString = "User Id=scott;Password=tiger;" +
            "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");
    }
}
Password Expiration

Oracle allows users passwords 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 example uses the `OracleConnection` `OpenWithNewPassword` method to connect with a new password of `panther`:

```csharp
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");
        }
    }
}
```

See Also: DBA Privilege "Supported Connection String Attributes" on page 5-79 for further information on privileged connections in the database.
Proxy Authentication

With proper setup in the database, proxy authentication enables middle-tier applications to control the security by preserving database user identities and privileges, and auditing actions taken on behalf of these users. This is accomplished by creating and using a proxy database user that connects and authenticates against the database on behalf of a database user (that is, the real user) or database users.

Proxy authentication can then be used to provide better scalability with connection pooling. When connection pooling is used in conjunction with proxy authentication, the proxy authenticated connections can be shared among different real users. This is because only the connection and session established for the proxy is cached. An additional session is created for the real user when a proxy authenticated connection is requested, but it will be destroyed appropriately when the proxy authenticated connection is placed back into the pool. This design enables the application to scale well without sacrificing security.

ODP.NET applications can use proxy authentication by setting the "Proxy UserId" and "Proxy Password" attributes in the connection string. The real user is specified by the "User Id" attribute. Optionally, to enforce greater security, the real user’s password can be provided through the "Password" connection string attribute. When using distributed transactions in conjunction with proxy authentication, the real user’s password is no longer optional, and it must be supplied.

The following example illustrates the use of ODP.NET proxy authentication:

```csharp
/* Log on as DBA (SYS or SYSTEM) that has CREATE USER privilege.
   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;

*/

class ProxyAuthenticationSample
{
    static void Main()
    {
```

Note:

- The OpenWithNewPassword method should be used only when the user password has expired, not for changing the password.
- If connection pooling is enabled, then invoking the OpenWithNewPassword method also clears the connection pool. This closes all idle connections created with the old password.

See Also: "OpenWithNewPassword" on page 5-108
 dynamic Distributed Transaction Enlistment

For those applications that dynamically enlist in distributed transactions through the
EnlistDistributedTransaction of the OracleConnection object, the
"enlist" connection string attribute must be set to a value of either "dynamic" or
"true". If "enlist=true", the connection enlists in a transaction when the Open
method is called on the OracleConnection object, if it is within the context of a
COM+ transaction or a System.Transactions. If not, the OracleConnection
object does not enlist in a distributed transaction, but it can later enlist explicitly using
the EnlistDistributedTransaction or the EnlistTransaction method. If
"enlist=false", the connection cannot enlist in the transaction.

For applications that cannot be rebuilt using "Enlist=dynamic", a registry string
value, named DynamicEnlistment, of type REG_SZ, should be created and set to 1
under HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version
where Assembly_Version is the full assembly version number of
Oracle.DataAccess.dll.

If ODP.NET is properly installed, there should already be registry string values such
as StatementCacheSize, TraceFileName, and so forth, under the same ODP.NET
key. Dynamic Enlistment can also be configured through an XML configuration file.
For details, see "Configuring Oracle Data Provider for .NET" on page 2-5.

If the DynamicEnlistment registry key is set to 0 (or if the registry entry does not
exist), it does not affect the application in any way. However, if DynamicEnlistment
is set to 1, "Enlist=false" is treated the same as "Enlist=dynamic", enabling
applications to enlist successfully through the EnlistDistributedTransaction
method without any code change. Having DynamicEnlistment set to 1 does not
affect OracleConnection objects that have "Enlist=true" or
"Enlist=dynamic" in the connection string.

See Also:

■ Oracle Database Advanced Application Developer's Guide 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 the ALTER USER statement

■ Oracle Database Security Guide section "Standard Auditing in a
Multitier Environment"
Connecting to Oracle Database

Features of Oracle Data Provider for .NET

Client Identifier and End-to-End Tracing

The client identifier is a predefined attribute from the Oracle application context namespace USERENV. It is similar to proxy authentication because it can enable tracking of user identities. However, client identifier does not require the creation of two sessions (one for the proxy user and another for the end user) as proxy authentication does. In addition, the client identifier does not have to be a database user. It can be set to any string. But most importantly, by using client identifier, ODP.NET developers can use application context and Oracle Label Security, and configure Oracle Virtual Private Database (VPD) more easily. To set the client identifier, ODP.NET applications can set the ClientId property on the OracleConnection object after opening a connection. If connection pooling is enabled, the ClientId is reset to null whenever a connection is placed back into the pool.

The client identifier can also be used for end-to-end application tracing. End-to-end tracing simplifies the process of diagnosing performance problems in multitier environments. In multitier environments, a request from an end client is routed to different database sessions by the middle tier making it difficult to track a client across different database sessions. End-to-end tracing uses the client identifier to uniquely trace a specific end-client through all tiers to the database server.

ODP.NET exposes the ActionName, ClientId, ClientInfo, and ModuleName write-only properties on the OracleConnection object. These properties correspond to the following end-to-end tracing attributes:

- **Action** - Specifies an action, such as an INSERT or UPDATE operation, in a module
- **ClientId** - Specifies an end user based on the logon ID, such as HR.HR
- **ClientInfo** - Specifies user session information
- **ModuleName** - Specifies a functional block, such as Accounts Receivable or General Ledger, of an application

See Also:

- "OracleConnection Properties" on page 5-76
- "End-to-End Application Tracing" in the Oracle Database Performance Tuning Guide
- Oracle Database Security Guide

Transparent Application Failover (TAF) Callback Support

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

**Note:** TAF is not supported in a .NET stored procedure.

TAF enables an application connection to automatically reconnect to another database instance if the connection gets severed. Active transactions roll back, but the new
Connecting to Oracle Database

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

With TAF, 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.

When a session fails over to another database, the NLS settings that were initially set on the original session are not carried over to the new session. Therefore, it is the responsibility of the application to set these NLS settings on the new session.

**TAF Notification**

Given the delays that failovers can cause, applications may wish to be notified by a TAF callback. ODP.NET supports the TAF callback function 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 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` object that is passed to the event handler.

**Registering an Event Handler for Failover**

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

```csharp
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";
```
The `Failover` event invokes only 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 11-9
- "OracleFailoverEventArgs Class" on page 11-2
ADO.NET 2.0 Features

Oracle Data Provider for .NET 10.2.0.2 or later supports Microsoft ADO.NET 2.0 APIs.

This section contains the following topics:

- About ADO.NET 2.0
- Base Classes and Provider Factory Classes
- Connection String Builder
- Data Source Enumerator
- Support for Code Access Security
- Support for Schema Discovery
- System.Transactions and Promotable Transactions
- Batch Processing Support
- ADO.NET 2.0 Only Classes and Class Members
- Bulk Copy Support

About ADO.NET 2.0

ADO.NET 2.0 is a Microsoft specification that provides data access features designed to work together for provider independence, increased component reuse, and application convertibility. Additional features make it easier for an application to dynamically discover information about the data source, schema, and provider.

**Note:** Using ODP.NET with Microsoft ADO.NET 2.0 requires ADO.NET 2.0- compliant ODP.NET.

**See Also:**

ADO.NET in the MSDN Library

Base Classes and Provider Factory Classes

With ADO.NET 2.0, data classes derive from the base classes defined in the `System.Data.Common` namespace. Developers can create provider-specific instances of these base classes using provider factory classes.

Provider factory classes allow generic data access code to access multiple data sources with a minimum of data source-specific code. This reduces much of the conditional logic currently used by applications accessing multiple data sources.

Using Oracle Data Provider for .NET, the `OracleClientFactory` class can be returned and instantiated, enabling an application to create instances of the following ODP.NET classes that inherit from the base classes:

<table>
<thead>
<tr>
<th>ODP.NET Classes</th>
<th>Inherited from ADO.NET 2.0 Base Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleClientFactory</td>
<td>DbProviderFactory</td>
</tr>
<tr>
<td>OracleCommand</td>
<td>DbCommand</td>
</tr>
<tr>
<td>OracleCommandBuilder</td>
<td>DbCommandBuilder</td>
</tr>
</tbody>
</table>
In general, applications still require Oracle-specific connection strings, SQL or stored procedure calls, and declare that a factory from Oracle.DataAccess.Client is used.

**See Also:** "OracleClientFactory Class" on page 7-10

**Connection String Builder**

The OracleConnectionStringBuilder class makes creating connection strings less error-prone and easier to manage.

Using this class, developers can employ a configuration file to provide the connection string and/or dynamically set the values though the key/value pairs. One example of a configuration file entry follows:

```xml
<configuration>
    <connectionStrings>
        <add name="Publications" providerName="Oracle.DataAccess.Client"
             connectionString="User Id=scott;Password=tiger;Data Source=inst1" />
    </connectionStrings>
</configuration>
```

Connection string information can be retrieved by specifying the connection string name, in this example, Publications. Then, based on the providerName, the appropriate factory for that provider can be obtained. This makes managing and modifying the connection string easier. In addition, this provides better security against string injection into a connection string.

**See Also:** "OracleConnectionStringBuilder Class" on page 7-10

**Data Source Enumerator**

The data source enumerator enables the application to generically obtain a collection of the Oracle data sources that the application can connect to.

**See Also:** "OracleDataSourceEnumerator Class" on page 7-36

---

**Table 3–3 (Cont.) ODP.NET Classes that Inherit from ADO.NET 2.0 Base Classes**

<table>
<thead>
<tr>
<th>ODP.NET Classes</th>
<th>Inherited from ADO.NET 2.0 Base Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleConnection</td>
<td>DbConnection</td>
</tr>
<tr>
<td>OracleConnectionStringBuilder</td>
<td>DbConnectionStringBuilder</td>
</tr>
<tr>
<td>OracleDataAdapter</td>
<td>DbDataAdapter</td>
</tr>
<tr>
<td>OracleDataReader</td>
<td>DbDataReader</td>
</tr>
<tr>
<td>OracleDataSourceEnumerator</td>
<td>DbDataSourceEnumerator</td>
</tr>
<tr>
<td>OracleException</td>
<td>DbException</td>
</tr>
<tr>
<td>OracleParameter</td>
<td>DbParameter</td>
</tr>
<tr>
<td>OracleParameterCollection</td>
<td>DbParameterCollection</td>
</tr>
<tr>
<td>OracleTransaction</td>
<td>DbTransaction</td>
</tr>
</tbody>
</table>
Support for Code Access Security

ODP.NET implements code access security through the `OraclePermission` class. This ensures that application code trying to access the database has the requisite permission to do so.

When a .NET assembly tries to access Oracle Database through ODP.NET, ODP.NET demands `OraclePermission`. The .NET runtime security system checks to see whether the calling assembly, and all other assemblies in the call stack, have `OraclePermission` granted to them. If all assemblies in the call stack have `OraclePermission` granted to them, then the calling assembly can access the database. If any one of the assemblies in the call stack does not have `OraclePermission` granted to it, then a security exception is thrown.

Configuring OraclePermission

The `DemandOraclePermission` configuration attribute is used to enable or disable `OraclePermission` demand for an ODP.NET API. The `DemandOraclePermission` value can be specified in the Windows registry or an individual application configuration file.

The following Windows registry key is used to configure the `DemandOraclePermission` configuration attribute:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version\DemandOraclePermission
```

Here `Assembly_Version` is the full assembly version number of `Oracle.DataAccess.dll`. The `DemandOraclePermission` key is of type `REG_SZ`. It can be set to either 1 (enabled) or 0 (disabled).

You can also enable `OraclePermission` demand for an individual application using its application configuration file. The following example enables the `DemandOraclePermission` property in an application configuration file:

```
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="DemandOraclePermission" value="1"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

An application or assembly can successfully access the database if `OraclePermission` has been added to the permission set associated with the assembly’s code group. A system administrator can modify the appropriate permission set manually or by using the Microsoft .NET configuration tool (`Mscorcfg.msc`).

Administrators may also use an appropriate .NET Framework Tool, such as the Code Access Security Policy Tool (`Caspol.exe`), to modify security policy at the machine, user, and enterprise levels for including `OraclePermission`.

`OracleConnection` makes security demands using the `OraclePermission` object when `OraclePermission` demand has been enabled using `DemandOraclePermission` configuration attribute. Application developers should make sure that their code has sufficient permission before using `OracleConnection`. 
Configuring OraclePermission for Web Applications with High or Medium Trust Levels

For Web applications operating under high or medium trust, OraclePermission needs to be configured in the appropriate web_TrustLevel.config file, so that the application does not encounter any security errors.

OraclePermission can be configured using the OraProvCfg tool. OraProvCfg.exe adds appropriate entries to the web_hightrust.config and web_mediumtrust.config files associated with the specified .NET framework version.

The following example illustrates using the OraProvCfg tool for configuring OraclePermission in a .NET 2.0 Web application:

```
```

On running the preceding command, the following entry is added to the web_hightrust.config and web_mediumtrust.config files under the ASP.NET permission set:

```
<IPermission class="Oracle.DataAccess.Client.OraclePermission, Oracle.DataAccess, Version=2.112.2.0, Culture=neutral, PublicKeyToken=89b483f429c47342" version="1" Unrestricted="true" />
```

OraProvCfg can also be used to remove these entries from the .config files when required. The following example illustrates this:

```
```

Configuring OraclePermission for Windows Applications Running in a Partial Trust Environment

For Windows applications operating in a partial trust environment, the OraclePermission entry should be specified under the appropriate permission set in the security.config file. The security.config file is available in the %windir%\Microsoft.NET\Framework\{version}\CONFIG folder.

The following example specifies the OraclePermission entry for a .NET 2.0 Windows application:

```
<IPermission class="Oracle.DataAccess.Client.OraclePermission, Oracle.DataAccess, Version=2.112.2.0, Culture=neutral, PublicKeyToken=89b483f429c47342" version="1" Unrestricted="true" />
```

Support for Schema Discovery

ADO.NET 2.0 exposes five different types of metadata collections through the OracleConnection.GetSchema API. This permits application developers to customize metadata retrieval on an individual-application basis, for any Oracle data
source. Thus, developers can build a generic set of code to manage metadata from multiple data sources.

The following types of metadata are exposed:

- **MetaDataCollections**
  A list of metadata collections that is available from the data source, such as tables, columns, indexes, and stored procedures.

- **Restrictions**
  The restrictions that apply to each metadata collection, restricting the scope of the requested schema information.

- **DataSourceInformation**
  Information about the instance of the database that is currently being used, such as product name and version.

- **DataTypes**
  A set of information about each data type that the database supports.

- **ReservedWords**
  Reserved words for the Oracle query language.

**See Also:** Appendix A, "Oracle Schema Collections"

**User Customization of Metadata**

ODP.NET provides a comprehensive set of database schema information. Developers can extend or customize the metadata that is returned by the `GetSchema` method on an individual application basis.

To do this, developers must create a customized metadata file and provide the file name to the application as follows:

1. Create a customized metadata file and put it in the `CONFIG` subdirectory where the .NET framework is installed. This is the directory that contains `machine.config` and the security configuration settings.
   
   This file must contain the entire set of schema configuration information, not just the changes. Developers provide changes that modify the behavior of the schema retrieval to user-specific requirements. For instance, a developer can filter out internal database tables and just retrieve user-specific tables.

2. Add an entry in the `app.config` file of the application, similar to the following, to provide the name of the metadata file, in name-value pair format.

   ```xml
   <oracle.dataaccess.client>
   <settings>
   <add name="MetaDataXml" value="CustomMetaData.xml" />
   </settings>
   </oracle.dataaccess.client>
   ``

   When the `GetSchema` method is called, ODP.NET checks the `app.config` file for the name of the customized metadata XML file. First, the `GetSchema` method searches for an entry in the file with a `name` element named after the provider, in this example, `oracle.dataaccess.client`. In this XML element, the value that corresponds to the `name MetaDataXml` is the name of the customized XML file, in this example, `CustomMetaData.xml`. 


If the metadata file is not in the correct directory, then the application loads the default metadata XML file, which is part of ODP.NET.

See Also: "GetSchema" on page 5-100

System.Transactions and Promotable Transactions

ODP.NET for .NET Framework 2.0 supports System.Transactions. A local transaction is created for the first connection opened in the System.Transactions scope to Oracle Database 11g release 1 (11.1), or higher. When a second connection is opened, this transaction is automatically promoted to a distributed transaction. This functionality provides enhanced performance and scalability.

Connections created within a transaction context, such as TransactionScope or ServicedComponent, can be established to different versions of Oracle Database. However, in order to enable the local transaction to be promotable, the following must be true:

- The first connection in the transaction context must be established to an Oracle Database 11g release 1(11.1) instance or higher.
- All connections opened within the transaction context must have the "Promotable Transaction" setting set to "promotable". If you try to open a subsequent connection in the same transaction context with the "Promotable Transaction" setting set to "local", an exception is thrown.
- Promoting local transactions requires Oracle Services for Microsoft Transaction Server 11.1.0.7.20, or higher. If this requirement is not met, then a second connection request in the same transaction context throws an exception.

Setting "local" as the value of "PromotableTransaction" in the registry, configuration file (machine/Web/application), or the "Promotable Transaction" connection string attribute allows only one connection to be opened in the transaction context, which is associated with a local transaction. Such local transactions cannot be promoted.

For applications connecting to a pre-Oracle Database 11g release 1 (11.1) instance, refer to "Local Transaction Support for Older Databases" on page 3-28. This section describes how ODP.NET behavior can be controlled using the "Promotable Transaction" setting.

If applications use System.Transactions, it is required that the "enlist" connection string attribute is set to either "true" (default) or "dynamic".

ODP.NET supports the following System.Transactions programming models for applications using distributed transactions.

- Implicit Transaction Enlistment Using TransactionScope
- Explicit Transaction Enlistment Using CommittableTransaction.
- Local Transaction Support for Older Databases

Implicit Transaction Enlistment Using TransactionScope

The TransactionScope class provides a mechanism to write transactional applications where the applications do not need to explicitly enlist in transactions.

To accomplish this, the application uses the TransactionScope object to define the transactional code. Connections created within this transactional scope will enlist in a local transaction that can be promoted to a distributed transaction.
Note:

If the first connection is opened to a pre-Oracle Database 11g release 1 (11.1) instance, then the connection enlists as a distributed transaction, by default.

You can optionally create the transaction as a local transaction by using the procedure described in "Local Transaction Support for Older Databases" on page 28. However, these transactions cannot be promoted to distributed transactions.

---

Note that the application must call the Complete method on the TransactionScope object to commit the changes. Otherwise, the transaction is aborted by default.

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

class psfTxnScope
{
    static void Main()
    {
        int retVal = 0;
        string providerName = "Oracle.DataAccess.Client";
        string constr =
            "User Id=scott;Password=tiger;Data Source=oracle;enlist=true";

        // Get the provider factory.
        DbProviderFactory factory = DbProviderFactories.GetFactory(providerName);

        try
        {
            // Create a TransactionScope object, (It will start an ambient
            // transaction automatically).
            using (TransactionScope scope = new TransactionScope())
            {
                // Create first connection object.
                using (DbConnection conn1 = factory.CreateConnection())
                {
                    // Set connection string and open the connection. this connection
                    // will be automatically enlisted in a promotable local transaction.
                    conn1.ConnectionString = constr;
                    conn1.Open();

                    // Create a command to execute the sql statement.
                    DbCommand cmd1 = factory.CreateCommand();
                    cmd1.Connection = conn1;
                    cmd1.CommandText = "insert into emp (empno, ename, job) values (1234, 'emp1', 'dev1')";

                    // Execute the SQL statement to insert one row in DB.
                    retVal = cmd1.ExecuteNonQuery();
                    Console.WriteLine("Rows to be affected by cmd1: {0}", retVal);
                }
            }
        }
    }
}```
// Close the connection and dispose the command object.
conn1.Close();
conn1.Dispose();
cmd1.Dispose();

// The Complete method commits the transaction. If an exception has
// been thrown or Complete is not called then the transaction is
// rolled back.
scope.Complete();
}
}

} catch (Exception ex)
{
    Console.WriteLine(ex.Message);
    Console.WriteLine(ex.StackTrace);
}


Explicit Transaction Enlistment Using CommittableTransaction

The instantiation of the CommittableTransaction object and the
EnlistTransaction method provides an explicit way to create and enlist in a
transaction.

Note that the application must call Commit or Rollback on the
CommittableTransaction object.

// C#

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

class psfEnlistTransaction
{
    static void Main()
    {
        int retVal = 0;
        string providerName = "Oracle.DataAccess.Client";
        string constr =
            @"User Id=scott;Password=tiger;Data Source=oracle;enlist=dynamic;"

        // Get the provider factory.
        DbProviderFactory factory = DbProviderFactories.GetFactory(providerName);;

        try
        {
            // Create a committable transaction object.
            CommittableTransaction cmtTx = new CommittableTransaction();

            // Open a connection to the DB.
            DbConnection conn1 = factory.CreateConnection();
            conn1.ConnectionString = constr;
            conn1.Open();

            // enlist the connection with the committable transaction.
            conn1.EnlistTransaction(cmtTx);
ADO.NET 2.0 Features

// Create a command to execute the sql statement.
DbCommand cmd1 = factory.CreateCommand();
cmd1.Connection = conn1;
cmd1.CommandText = @"insert into emp (empno, ename, job) values
(1234, 'emp1', 'dev1')";
// Execute the SQL statement to insert one row in DB.
retVal = cmd1.ExecuteNonQuery();
Console.WriteLine("Rows to be affected by cmd1: {0}", retVal);
// commit/rollback the transaction.
cmtTx.Commit();
// commits the txn.
//cmtTx.Rollback(); // rolls back the txn.
// close and dispose the connection
conn1.Close();
conn1.Dispose();
cmd1.Dispose();
}
catch (Exception ex)
{
Console.WriteLine(ex.Message);
Console.WriteLine(ex.StackTrace);
}
}
}

See Also:

"EnlistTransaction" on page 5-98

Local Transaction Support for Older Databases
If the first connection in a TransactionScope is opened to a pre-Oracle Database
11g release 1 (11.1) instance, then the connection creates a distributed transaction, by
default. You can optionally have the fist connection create a local transaction by using
the procedure described in this section.
To create local transactions in a System.Transactions scope, either the
PromotableTransaction setting in the registry, machine/Web/application
configuration file, or the "Promotable Transaction" connection string attribute
must be set to "local".
If "local" is specified, the first connection opened in the TransactionScope uses a
local transaction. If any subsequent connections are opened within the same
TransactionScope, an exception is thrown. If there are connections already opened
in the TransactionScope, and an OracleConnection with "Promotable
Transaction=local" attempts to open within the same TransactionScope, an
exception is thrown.
If "promotable" is specified, the first and all subsequent connections opened in the
same TransactionScope enlist in the same distributed transaction.
If both the registry and the connection string attribute are used and set to different
values, the connection string attribute overrides the registry entry value. If neither are
set, "promotable" is used. This is the default value and is equivalent to previous
versions of ODP.NET which only supported distributed transactions.
The registry entry for a particular version of ODP.NET applies for all applications
using that version of ODP.NET.

3-28 Oracle Data Provider for .NET Developer's Guide


Batch Processing Support

The `OracleDataAdapter` `UpdateBatchSize` property enables batch processing when the `OracleDataAdapter.Update` method is called. `UpdateBatchSize` is a numeric property that indicates how many DataSet rows to update the Oracle database for each round-trip.

This enables the developer to reduce the number of round-trips to the database.

---

**Note:** Microsoft Hotfix Needed

There is a known issue in Microsoft ADO.NET 2.0 that affects the `BatchUpdate` functionality.

To resolve this issue, both ODP.NET release 11.1 and a specific Microsoft hotfix must be installed on the same computer. The Microsoft hotfix is available for free download from the following site: [http://support.microsoft.com/?id=916002](http://support.microsoft.com/?id=916002)

Without this fix, the `BatchUpdate` feature does not provide the correct error description for the failed rows in the `DataSet`. All errors in a batch are either appended to the exception message, if `DbDataDataAdapter.ContinueUpdateOnError` is `false`, or appended to the `RowError` property of the last updated row of the `DataSet`.

ODP.NET has been enhanced to use this hotfix and to populate the correct error description to the `RowError` property of the individual failed rows in a batch.

---

**See Also:** "UpdateBatchSize" on page 5-129

**ADO.NET 2.0 Only Classes and Class Members**

In addition to classes which are ADO.NET 2.0 only, other ODP.NET classes that inherit from the `System.Data.Common` namespace include methods and properties which require ADO.NET 2.0.

The following classes are ADO.NET 2.0 only:

- **OracleClientFactory Class**
- **OracleConnectionStringBuilder Class**
- **OracleDataSourceEnumerator Class**

The following class members are ADO.NET 2.0 only:

- **OracleCommandBuilder Class Members**
  - **CatalogLocation** Property *(Not Supported)*
  - **CatalogSeparator** Property *(Not Supported)*
  - **ConflictOption** Property *(Not Supported)*
  - **QuotePrefix** Property
  - **QuoteSuffix** Property
  - **SchemaSeparator** Property
  - **QuoteIdentifier** Method
ADO.NET 2.0 Features

- **OracleConnection** Class Members
  - UnquoteIdentifier Method
- **OracleDataAdapter** Class Members
  - GetSchema Methods
- **OracleDataReader** Class Members
  - HiddenFieldCount Property
  - VisibleFieldCount Property
  - GetProviderSpecificFieldType Method
  - GetProviderSpecificValue Method
  - GetProviderSpecificValues Method
- **OracleParameter** Class Members
  - SourceColumnNullMapping Property
  - ResetDbType Method
  - ResetOracleDbType Method
- **OracleParameterCollection** Class Members
  - AddRange Method

**Bulk Copy Support**

ODP.NET provides a Bulk Copy feature which enables applications to efficiently load large amounts of data from a table in one database to another table in the same or a different database.

The ODP.NET Bulk Copy feature uses a direct path load approach, which is similar to, but not the same as Oracle SQL*Loader. Using direct path load is faster than conventional loading (using conventional SQL INSERT statements). Conventional loading formats Oracle data blocks and writes the data blocks directly to the data files. Bulk Copy eliminates considerable processing overhead.

The ODP.NET Bulk Copy feature can load data into older Oracle databases.

**See Also:** "System Requirements" on page 2-1 to learn which versions of the Oracle Database ODP.NET interoperates with

The ODP.NET Bulk Copy feature is subject to the same basic restrictions and integrity constraints for direct path loads, as discussed in the next few sections.

**Data Types Supported by Bulk Copy**

The data types supported by Bulk Copy are:

- ORA_SB4
- ORA_VARNUM
- ORA_FLOAT
- ORA_CHARN
ADO.NET 2.0 Features

Features of Oracle Data Provider for .NET

- ORA_RAW
- ORA_BFLOAT
- ORA_BDOUBLE
- ORA_IBDOUBLE
- ORA_IBFLOAT
- ORA_DATE
- ORA_TIMESTAMP
- ORA_TIMESTAMP_TZ
- ORA_TIMESTAMP_LTZ
- ORA_INTERVAL_DS
- ORA_INTERVAL_YM

Bulk copy does not support overwrites.

Restrictions on Oracle Bulk Copy of a Single Partition
- The table that contains the partition cannot have any global indexes defined on it.
- The tables that the partition is a member of cannot have referential and check constraints enabled.
- Enabled triggers are not allowed.

Integrity Constraints Affecting Oracle Bulk Copy
During a Oracle bulk copy, some integrity constraints are automatically enabled or disabled, as follows:

Enabled Constraints
During an Oracle bulk copy, the following constraints are automatically enabled by default:
- NOT NULL
- UNIQUE
- PRIMARY KEY (unique-constraints on not-null columns)

NOT NULL constraints are checked at column array build time. Any row that violates the NOT NULL constraint is rejected.

UNIQUE constraints are verified when indexes are rebuilt at the end of the load. The index is left in an Index Unusable state if it violates a UNIQUE constraint.

Disabled Constraints
During an Oracle bulk copy, the following constraints are automatically disabled by default:
- CHECK constraints
- Referential constraints (FOREIGN KEY)

If the EVALUATE CHECK_CONSTRAINTS clause is specified, then CHECK constraints are not automatically disabled. The CHECK constraints are evaluated during a direct path load and any row that violates the CHECK constraint is rejected.
Database Insert Triggers
Table insert triggers are disabled when a direct path load begins. After the rows are loaded and indexes rebuilt, any triggers that were disabled are automatically reenabled. The log file lists all triggers that were disabled for the load. There should be no errors reenabling triggers.

Unlike integrity constraints, insert triggers are not reapplied to the whole table when they are enabled. As a result, insert triggers do not fire for any rows loaded on the direct path. When using the direct path, the application must ensure that any behavior associated with insert triggers is carried out for the new rows.

Field Defaults
Default column specifications defined in the database are not available with direct path loading. Fields for which default values are desired must be specified with the `DEFAULTIF` clause. If a `DEFAULTIF` clause is not specified and the field is `NULL`, then a null value is inserted into the database.

See Also: Chapter 17, "Oracle Data Provider for .NET Bulk Copy Classes"

ADO.NET Entity Framework and LINQ to Entities

ODP.NET 11.2.0.3.0, and higher, includes support for the ADO.NET Entity Framework and LINQ to Entities. ODP.NET also supports Entity SQL.

Entity Framework is a framework for providing object-relational mapping service on data models. Entity Framework addresses the impedance mismatch between the relational database format and the client’s preferred object format. Language Integrated Query (LINQ) defines a set of operators that can be used to query, project, and filter data in arrays, enumerable classes, XML, relational databases, and other data sources. One form of LINQ, LINQ to Entities, allows querying of Entity Framework data sources. ODP.NET supports Entity Framework such that the Oracle database can participate in object-relational modeling and LINQ to Entities queries.

Entity Framework and LINQ provides productivity benefits for the .NET developer. It abstracts the database’s data model from the application’s data model. Working with object-relational data becomes easier with Entity Framework’s tools. Oracle’s integration with Entity Framework and LINQ enables Oracle .NET developers to take advantage of all these productivity benefits.

Note:

- Entity Framework and LINQ to Entities support is included in ODP.NET for .NET Framework 4. ODP.NET for .NET Framework 2.0 does not support the ADO.NET Entity Framework and LINQ to Entities.
- Entity Framework 4.1 is supported. However, the Code First feature, that is part of Entity Framework 4.1, is not currently supported.
- Binding scalar parameters is supported with ODP.NET and Entity Framework. In Entity Framework, parameter binding by name is supported. Binding by position is not supported.
Entity data models can now be generated from Oracle database schemas. These Oracle entity data models can be queried and manipulated using Visual Studio and ODP.NET. Oracle supports Database First and Model First modeling approaches. Specifying filters on the Visual Studio Server Explorer data connection enables the Entity Data Model Wizard to also filter Oracle database objects that are fetched and displayed.

LINQ to Entities can perform queries on the Oracle Database using ODP.NET, including using LINQ to Entities built-in functions. INSERTS, UPDATES, and DELETES can be executed using Oracle stored procedures, or by using the ObjectContext SaveChanges method.

ODP.NET supports function import of Oracle stored procedures that Entity Framework can then execute. These Oracle function imports can return a collection of scalar, complex, and entity types, including returning an Oracle implicit result set as an entity type. Implicit result set binding is supported using Oracle REF CURSOR. See "Implicit REF CURSOR Binding" on page 3-75 for more details.

See Also: For a tutorial on how to use Entity Framework, Language Integrated Query (LINQ), and generate Data Definition Language (DDL) scripts using Model First, refer to:


### Mapping Oracle Data Types to EDM Types

The ODP.NET manifest file describes the primitive types, such as VARCHAR2 and Number, and the Entity Data Model (EDM) types, such as string and Int32, that they map to. It also includes the facets for each EDM type.

ODP.NET does not support Time literals and canonical functions related to the Time type.

Oracle considers both NULL and empty strings to be NULL strings and are considered to be equal. Operations, such as Equals(), Length(), and Trim() on such strings will result in a NULL string.

Table 3–4 maps the Oracle data types to their corresponding EDM types. The table also includes details about provider type attributes and the EDM type facets associated with each Oracle data type.

#### Table 3–4 Mapping of Oracle Data Types and EDM Types

<table>
<thead>
<tr>
<th>Oracle Data Types</th>
<th>EDM Types (Primitive-TypeKind)</th>
<th>Provider Type Attributes: Name and Value</th>
<th>EDM Type Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bfile</td>
<td>Binary</td>
<td>• Equal Comparable: False</td>
<td>EDM Type Facets for Bfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Order Comparable: False</td>
<td></td>
</tr>
<tr>
<td>Binary_Double</td>
<td>Double</td>
<td>• Equal Comparable: True</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>(introduced in 10g)</td>
<td></td>
<td>• Order Comparable: True</td>
<td></td>
</tr>
</tbody>
</table>
### Table 3-4  (Cont.) Mapping of Oracle Data Types and EDM Types

<table>
<thead>
<tr>
<th>Oracle Data Types</th>
<th>EDM Types (Primitive-TypeKind)</th>
<th>Provider Type Attributes: Name and Value</th>
<th>EDM Type Facets</th>
</tr>
</thead>
</table>
| Binary_Float (introduced in 10g) | Single | • Equal Comparable: True  
• Order Comparable: True | Not Applicable |
| Binary_Integer | Int32 | • Equal Comparable: True  
• Order Comparable: True | Not Applicable |
| Blob | Binary | • Equal Comparable: False  
• Order Comparable: False | EDM Type Facets for Blob |
| Char | String | • Equal Comparable: True  
• Order Comparable: True | EDM Type Facets for Char |
| Clob | String | • Equal Comparable: False  
• Order Comparable: False | EDM Type Facets for Clob |
| Date | DateTime | • Equal Comparable: True  
• Order Comparable: True | EDM Type Facets for Date |
| Float | Decimal | • Equal Comparable: True  
• Order Comparable: True | EDM Type Facets for Float |
| Int | Int32 | • Equal Comparable: True  
• Order Comparable: True | Not Applicable |
| Interval Day To Second | Decimal | • Equal Comparable: True  
• Order Comparable: True | EDM Type Facets for Interval Day To Second |
| Interval Year To Month | Decimal | • Equal Comparable: True  
• Order Comparable: True | EDM Type Facets for Interval Year To Month |
| Long | String | • Equal Comparable: False  
• Order Comparable: False | EDM Type Facets for Long |
<table>
<thead>
<tr>
<th>Oracle Data Types</th>
<th>EDM Types (Primitive-TypeKind)</th>
<th>Provider Type Attributes: Name and Value</th>
<th>EDM Type Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long Raw</td>
<td>Binary</td>
<td>Equal Comparable: False</td>
<td>EDM Type Facets for Long Raw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: False</td>
<td></td>
</tr>
<tr>
<td>NChar</td>
<td>String</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for NChar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>NClob</td>
<td>String</td>
<td>Equal Comparable: False</td>
<td>EDM Type Facets for NClob</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: False</td>
<td></td>
</tr>
<tr>
<td>Nested Table</td>
<td>Not Applicable</td>
<td>Not Applicable and Not Supported</td>
<td></td>
</tr>
<tr>
<td>Number(1,0)</td>
<td>Int16</td>
<td>Equal Comparable: True</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Number(2,0)</td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Number(3,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(4,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(5,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(6,0)</td>
<td>Int32</td>
<td>Equal Comparable: True</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Number(7,0)</td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Number(8,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(9,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(10,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(11,0)</td>
<td>Int64</td>
<td>Equal Comparable: True</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Number(12,0)</td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Number(13,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(14,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(15,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(16,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(17,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(18,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number(19,0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number (all other cases)</td>
<td>Decimal</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>NVarchar2</td>
<td>String</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for NVarchar2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td>Not Applicable</td>
<td>Not Applicable and Not Supported</td>
<td></td>
</tr>
</tbody>
</table>

Features of Oracle Data Provider for .NET 3-35
### Table 3–4 (Cont.) Mapping of Oracle Data Types and EDM Types

<table>
<thead>
<tr>
<th>Oracle Data Types</th>
<th>EDM Types (Primitive-TypeKind)</th>
<th>Provider Type Attributes: Name and Value</th>
<th>EDM Type Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>Binary</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for Raw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Raw(16)</td>
<td>Guid</td>
<td>Equal Comparable: True</td>
<td>Not Applicable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Ref</td>
<td>Not Applicable</td>
<td>Not Applicable and Not Supported</td>
<td></td>
</tr>
<tr>
<td>ROWID</td>
<td>String</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for ROWID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Smallint</td>
<td>Int16</td>
<td>Equal Comparable: True</td>
<td>Not Applicable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>DateTime</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for Timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>DateTime</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for Timestamp with Local Time Zone</td>
</tr>
<tr>
<td>with</td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Local Time Zone</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>DateTimeOffset</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for Timestamp with Time Zone</td>
</tr>
<tr>
<td>with</td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Time Zone</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UROWID</td>
<td>Binary</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for UROWID</td>
</tr>
<tr>
<td>(size)</td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>Varchar2</td>
<td>String</td>
<td>Equal Comparable: True</td>
<td>EDM Type Facets for Varchar2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Order Comparable: True</td>
<td></td>
</tr>
<tr>
<td>VArray</td>
<td>Not Applicable</td>
<td>Not Applicable and Not Supported</td>
<td></td>
</tr>
<tr>
<td>XMLType</td>
<td>String</td>
<td>Equal Comparable: False</td>
<td>EDM Type Facets for XMLType</td>
</tr>
<tr>
<td>(introduced in 9i)</td>
<td></td>
<td>Order Comparable: False</td>
<td></td>
</tr>
</tbody>
</table>
EDM Type Facets
The following sections enumerate the EDM type facets for the preceding Oracle data types:

EDM Type Facets for Bfile

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>DefaultValue</td>
<td>2147483648</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

EDM Type Facets for Blob

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>DefaultValue</td>
<td>2147483648</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

EDM Type Facets for Char

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>Minimum</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
<tr>
<td>Unicode</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

EDM Type Facets for Clob

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>DefaultValue</td>
<td>2147483647</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>Unicode</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

EDM Type Facets for Date
<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>0</td>
</tr>
</tbody>
</table>

**EDM Type Facets for Float**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
<tr>
<td>Scale</td>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
</tbody>
</table>

**EDM Type Facets for Interval Day To Second**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Minimum</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>251</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>251</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
<tr>
<td>Scale</td>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
</tbody>
</table>

**Note:** EDM types do not support `TimeSpan`. Use `Decimal` to represent the total number of seconds. An application can obtain a `TimeSpan` by using the `TimeSpan.FromSeconds` static method.

**EDM Type Facets for Interval Year To Month**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Minimum</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
<tr>
<td>Facet name</td>
<td>Attribute name</td>
<td>Value</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-------</td>
</tr>
<tr>
<td>Scale</td>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
</tbody>
</table>

**EDM Type Facets for Long**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>DefaultValue</td>
<td>2147483647</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>Unicode</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

**EDM Type Facets for Long Raw**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>DefaultValue</td>
<td>2147483647</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

**EDM Type Facets for NChar**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>Minimum</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
<tr>
<td>Unicode</td>
<td>DefaultValue</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

**Note:** For NChar, the actual data is subject to the maximum byte limit of 2000.

The value of 1000 for Maximum and DefaultValue allows the EDM wizard to display columns of NCHAR(1000), where 1000 is the maximum number of characters allowed in DDL.

**EDM Type Facets for NClob**
ADO.NET Entity Framework and LINQ to Entities

### EDM Type Facets for Number

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>DefaultValue</td>
<td>2147483647</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>Unicode</td>
<td>DefaultValue</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

### EDM Type Facets for NVarchar2

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>Minimum</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
<tr>
<td>Scale</td>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
<tr>
<td>Unicode</td>
<td>DefaultValue</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

**Note:** For NVARCHAR2, the actual data is subject to the maximum byte limit of 4000.

The value of 2000 for Maximum and DefaultValue allows the EDM wizard to display columns of NVARCHAR2(2000), where 2000 is the maximum number of characters allowed in DDL.

### EDM Type Facets for Raw
<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>Minimum</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

**EDM Type Facets for ROWID**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxLength</td>
<td>DefaultValue</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>Unicode</td>
<td>DefaultValue</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
<tr>
<td>FixedLength</td>
<td>DefaultValue</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>True</td>
</tr>
</tbody>
</table>

**EDM Type Facets for Timestamp**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
</tbody>
</table>

**EDM Type Facets for Timestamp with Local Time Zone**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
</tbody>
</table>

**EDM Type Facets for Timestamp with Time Zone**

<table>
<thead>
<tr>
<th>Facet name</th>
<th>Attribute name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Constant</td>
<td>False</td>
</tr>
</tbody>
</table>

**EDM Type Facets for UROWID**
Oracle Number Default Data Type Mapping and Customization

You can configure a custom mapping in the .NET configuration file to override the default mapping for the \texttt{Number(p,0)} Oracle data type. So, for example, \texttt{Number(1,0)}, which is mapped to \texttt{Int16} by default, can be custom mapped to the .NET \texttt{Bool} or .NET \texttt{Byte} type.

\textbf{Example 3–1} shows a sample \texttt{app.config} file that uses custom mapping to map the \texttt{Number(1, 0)} Oracle data type to the \texttt{bool} EDM type. The example also maps \texttt{Number(3,0)} to \texttt{byte}, and sets the maximum precisions for the \texttt{Int16}, \texttt{Int32}, and \texttt{Int64} data types to 4, 9, and 18 respectively.

\textbf{Example 3–1 \ Sample Application Configuration File to Custom Map the Number (p,0) Data Type}

```xml
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
    <!-- Other connection strings go here -->
  </connectionStrings>
  <oracle.dataaccess.client>
    <Facet name="MaxLength" Attribute name="DefaultValue" Value="4000"/>
    <Facet name="MaxLength" Attribute name="Constant" Value="True"/>
    <Facet name="FixedLength" Attribute name="DefaultValue" Value="True"/>
    <Facet name="FixedLength" Attribute name="Constant" Value="True"/>
  </oracle.dataaccess.client>
</configuration>
```
Example 3–1 customizes the mappings as follows:

<table>
<thead>
<tr>
<th>Oracle Type</th>
<th>Default EDM Type</th>
<th>Custom EDM Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number(1, 0)</td>
<td>Int16</td>
<td>bool</td>
</tr>
<tr>
<td>Number(2, 0) to Number(3, 0)</td>
<td>Int16</td>
<td>byte</td>
</tr>
<tr>
<td>Number(4, 0)</td>
<td>Int16</td>
<td>Int16</td>
</tr>
<tr>
<td>Number(5, 0)</td>
<td>Int16</td>
<td>Int32</td>
</tr>
<tr>
<td>Number(6, 0) to Number(9, 0)</td>
<td>Int32</td>
<td>Int32</td>
</tr>
<tr>
<td>Number(10, 0)</td>
<td>Int32</td>
<td>Int64</td>
</tr>
<tr>
<td>Number(11, 0) to Number(18, 0)</td>
<td>Int64</td>
<td>Int64</td>
</tr>
<tr>
<td>Number(19, 0)</td>
<td>Int64</td>
<td>Decimal</td>
</tr>
</tbody>
</table>

Custom mapping configures the maximum precision of the Oracle Number type that would map to the .NET/EDM type. So, for example, the preceding custom application configuration file configures ODP.NET to map Number(10, 0) through Number(18, 0) to Int64, as opposed to the default range of Number(11, 0) through Number(19, 0) for Int64.

**Note:**

- Custom mapping does not require you to map all the .NET/EDM types. For example, if custom mapping is required just for Int16, then having a single entry for Int16 is sufficient. Default mapping gets used for the other types.
- When using Model First, a Byte attribute is mapped to Number(3, 0) by default. However, when a model is generated for a Number(3, 0) column, it gets mapped to Int16 by default unless custom mapping for Byte is specified.

You must make sure that your mappings allow the data to fit within the range of the .NET/EDM type and the Number(p, s) type. If you select a .NET/EDM type with a range too small for the Oracle Number data, then errors will occur during data retrieval. Also, if you select a .NET/EDM type, and the corresponding data is too big for the Oracle Number column, then INSERTs and UPDATEs to the Oracle database will error out.
Resolving Compilation Errors When Using Custom Mapping
If the custom mapping in a .NET configuration file has changed, then regenerate the
data model to solve compilation errors introduced by the changes.

Under certain scenarios, custom mapping may cause compilation errors when a
project that uses custom mapping is loaded by Visual Studio. You may use the
following workaround for such scenarios:

1. Open Visual Studio Help, About Microsoft Visual Studio. Click OK to exit the
dialog box.
   Alternatively, open the to-be-used connection in Server Explorer.
2. Compile the project again to eliminate the compilation errors.

Mapping Boolean and Guid Parameters in Custom INSERT, UPDATE, and DELETE
Stored Procedures
When using your custom INSERT, UPDATE, or DELETE stored procedure in Stored
Procedure Mapping, the following error might occur:
Error 2042: Parameter Mapping specified is not valid.
This can happen if a Number parameter has been mapped to a Boolean attribute, or if
a RAW parameter has been mapped to a Guid attribute.
The solution is to manually add Precision="1" for the Number parameter, and
MaxLength="16" for the RAW parameter of your stored procedure in the SSDL.

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

Note: Optimizer hint syntax in the form ++ ... is not supported.
ODP.NET supports this syntax: /*+ ... */.

This section includes the following topics:

- Transactions
- Parameter Binding
- Statement Caching
- Self-Tuning

Transactions
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 be started only on an
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.

Note: Transactions are not supported in a .NET stored procedure.
Explicit transactions are required with SQL statements containing "FOR UPDATE" and "RETURNING" clauses. This is not necessary if global transactions are used.

**Parameter Binding**

When the DbType property of an OracleParameter object is set, the OracleDbType property of the OracleParameter object changes accordingly, or vice versa. The parameter set last prevails.

An application can bind the data and have ODP.NET infer both the DbType and OracleDbType properties from the .NET type of the parameter value.

ODP.NET allows applications to obtain an output parameter as either a .NET Framework type or an ODP.NET type. The application can specify which type to return for an output parameter by setting the DbType property of the output parameter (.NET type) or the OracleDbType property (ODP.NET type) of the OracleParameter object. For example, if the output parameter is set as a String type by setting the DbType property, the output data is returned as a .NET String type. On the other hand, if the parameter is set as an OracleDbType.Char type by setting the OracleDbType property, the output data is returned as an OracleString type. If both DbType and OracleDbType properties are set before the command execution, the last setting takes affect.

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.

When binding by position (default) to a function, ODP.NET expects the return value to be bound first, before any other parameters.

This section describes the following:

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

See Also: "OracleDbType Enumeration" on page 5-360

**Data Types BINARY_FLOAT and BINARY_DOUBLE**

Starting from Oracle Database 10g, the database supports two new native data types, BINARY_FLOAT and BINARY_DOUBLE.

The BINARY_FLOAT and BINARY_DOUBLE data types represent single-precision and double-precision, floating-point values respectively.

In OracleParameter binding, an application should use the enumerations OracleDbType.BinaryFloat and OracleDbType.BinaryDouble for BINARY_FLOAT and BINARY_DOUBLE data types.
OracleDbType Enumeration Type

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

Table 3–5 lists all the OracleDbType enumeration values with a description of each enumerated value.

### Table 3–5 OracleDbType Enumeration Values

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Oracle Collection type</td>
</tr>
<tr>
<td>BFile</td>
<td>Oracle BFILE type</td>
</tr>
<tr>
<td>BinaryFloat</td>
<td>Oracle BINARY_FLOAT type</td>
</tr>
<tr>
<td>BinaryDouble</td>
<td>Oracle BINARY_DOUBLE 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>Object</td>
<td>Oracle Object 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>Ref</td>
<td>Oracle REF 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>
</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 3–6.

<table>
<thead>
<tr>
<th>OracleDbType</th>
<th>System.Data.DbType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Object</td>
</tr>
<tr>
<td>BFile</td>
<td>Object</td>
</tr>
<tr>
<td>Blob</td>
<td>Object</td>
</tr>
<tr>
<td>BinaryFloat</td>
<td>Single</td>
</tr>
<tr>
<td>BinaryDouble</td>
<td>Double</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>
<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>Object</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>
</tbody>
</table>

**Note:** PL/SQL LONG, LONG RAW, RAW, and VARCHAR data types can be bound with a size up to 32512 bytes.
Inference of OracleDbType from DbType

In the `OracleParameter` class, specifying the value of `DbType` infers the value of `OracleDbType` as shown in Table 3-7.

### Table 3-7: Inference of OracleDbType from DbType

<table>
<thead>
<tr>
<th>OracleDbType</th>
<th>System.Data.DbType</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVarchar2</td>
<td>String</td>
</tr>
<tr>
<td>Object</td>
<td>Object</td>
</tr>
<tr>
<td>Raw</td>
<td>Binary</td>
</tr>
<tr>
<td>Ref</td>
<td>Object</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 (Cont.)

Table 3-6 (Cont.) Inference of System.Data.DbType from OracleDbType

<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>
<tr>
<td>Int64</td>
<td>Int64</td>
</tr>
<tr>
<td>Object</td>
<td>Object</td>
</tr>
<tr>
<td>Sbyte</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Single</td>
<td>Single</td>
</tr>
<tr>
<td>String</td>
<td>Varchar2</td>
</tr>
<tr>
<td>StringFixedLength</td>
<td>Char</td>
</tr>
<tr>
<td>Time</td>
<td>TimeStamp</td>
</tr>
<tr>
<td>UInt16</td>
<td>Not Supported</td>
</tr>
<tr>
<td>UInt32</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Uint64</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>
Inference of DbType and OracleDbType from Value  In the `OracleParameter` class, `Value` is an object type that can be of any .NET Framework data type or ODP.NET type. If the `OracleDbType` and `DbType` properties of the `OracleParameter` class are not specified, the `OracleDbType` property is inferred from the type of the `Value` property.

Table 3–8 shows the inference of `DbType` and `OracleDbType` properties from the `Value` property when the type of `Value` is one of the .NET Framework data types.

### Table 3–8  Inference of DbType and OracleDbType from Value (.NET Datatypes)

<table>
<thead>
<tr>
<th>Value (.NET Data types)</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>IOracleCustomType</td>
<td>Object</td>
<td>Object</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>Object</td>
<td>IntervalDS</td>
</tr>
</tbody>
</table>

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

Table 3–9 shows the inference of `DbType` and `OracleDbType` properties from the `Value` property when type of `Value` is one of `Oracle.DataAccess.Types`.

### Table 3–9  Inference of DbType and OracleDbType from Value (ODP.NET 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>
</tbody>
</table>
ODP.NET supports PL/SQL Associative Arrays (formerly known as PL/SQL Index-By Tables) binding.

An application can bind an OracleParameter object, 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 be set to an array of values, null, or the DBNull.Value property.
ODP.NET supports binding parameters of PL/SQL Associative Arrays which contain the following data types.

- BINARY_FLOAT
- CHAR
- DATE
- NCHAR
- NUMBER
- NVARCHAR2
- RAW
- ROWID
- UROWID
- VARCHAR2

Using unsupported data types with associative arrays can cause an ORA-600 error.

**Example of PL/SQL Associative Arrays**

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**

```sql
/* Setup the tables and required PL/SQL:

    connect scott/tiger@oracle
    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
        Param3(1) := Param2(1);
        Param3(2) := NULL;
        Param3(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
            -- ...
        END LOOP;
    END TestVarchar2;
END MYPACK;
```


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();
        Console.WriteLine("Connected to Oracle" + con.ServerVersion);

        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
            "First Element", "Second Element ", "Third Element "
        };
            "First Element", "Second Element ", "Third Element "
        };
        Param3.Value = null;

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

        // Setup the ArrayBindSize for Param1

        // Setup the ArrayBindStatus for Param1
            OracleParameterStatus.Success, OracleParameterStatus.Success,
            OracleParameterStatus.Success};
// 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
con.Close();
con.Dispose();
Console.WriteLine("Disconnected");

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

// C#

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
OracleCommand Object

```csharp
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();
```

See Also: "Value" on page 5-277 for more information

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

- **ArrayBindSize**
  
  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` object, except the `ArrayBindSize` property specifies the size for each value in an array.

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

  The `ArrayBindSize` property is used only for parameter types that have variable length such as `Clob`, `Blob`, and `Varchar2`. The size is represented in bytes for binary data types, 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 data types.

- **ArrayBindStatus**

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

  Before the execution, the application must populate the `ArrayBindStatus` property. After the execution, ODP.NET populates the property. Before the execution, an application using the `ArrayBindStatus` property can specify a NULL value for the corresponding element in the array for a parameter. After the execution, ODP.NET populates the `ArrayBindStatus` property, 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` object is thrown during an array bind execution, the `OracleErrorCollection` object 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.

The following example demonstrates error handling for array binding:

```csharp
/* Database Setup
connect scott/tiger@oracle
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[3] { 10, 200000, 30 };
            // int[] myArrayDeptNo = new int[3]{ 10,20,30};

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

            // 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 ex)
        {
            // Get the OracleErrorCollection
            OracleErrorCollection errors = ex.GetOracleErrorCollection();

            // Loop through the errors to find the one that occurred on the 3rd row
            foreach (OracleError error in errors)
            {
                if (error.ArrayBindIndex == 3)
                {
                    // The error occurred on the 3rd row
                    Console.WriteLine("Error occurred on row 3: ", error.Message);
                    break;
                }
            }
        }
    }
}
```
try{
    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 5-220 for more information

OracleParameterStatus Enumeration Types  Table 3–10 lists OracleParameterStatus enumeration values.

<table>
<thead>
<tr>
<th>Member Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>For input parameters, indicates that the input value has been assigned to the column. For output parameters, 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 execution by caching server cursors created during the initial statement execution. Subsequent executions of the same statement can reuse the parsed information from the cursor, and then execute the statement without reparsing, for better performance.

In order to see performance gains from statement caching, Oracle recommends caching only those statements that will be repeatedly executed. Furthermore, SQL or PL/SQL statements should use parameters rather than literal values. Doing so takes full advantage of statement caching, because parsed information from parameterized statements can be reused even if the parameter values change in subsequent executions. However, if the literal values in the statements are different, the parsed information cannot be reused unless the subsequent statements also have the same literal 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 (disabled). If it is set to a value greater than 0, ODP.NET statement caching is enabled and the value specifies the 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.

  If self tuning is enabled, then statement caching is enabled as well. The Statement Cache Size is configured automatically in such cases.

- **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.

Enabling Statement Caching through the Registry
To enable statement caching by default for all ODP.NET applications running in a system, without changing the application, set the registry key of `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version\StatementCacheSize` to a value greater than 0. This value specifies the number of cursors that are to be cached on the server.

The default value for the system can be overridden at the connection pool level. The Statement Cache Size attribute can be set to a different size than the registry value or it can be turned off. The Statement Cache Size can also be configured through an XML configuration file. For details, see “Configuring Oracle Data Provider for .NET” on page 2-5.

Statement Caching Methods and Properties
The following property and method are relevant only when statement caching is enabled:

- **OracleCommand.AddToStatementCache property**
  If statement caching is enabled, having this property set to `true` (default) 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.

- **OracleConnection.PurgeStatementCache method**
  This method purges all the cached statements by closing all open cursors on the database 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 the Statement Cache Purge attribute is set to false, statements executed on each separate connection are cached throughout the lifetime of the pooled connection.

If the Statement Cache Purge attribute 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 of.

Self-Tuning

ODP.NET applications can be self-tuned for performance optimization. ODP.NET dynamically monitors application queries during runtime.

---

**Note:** Self-tuning for applications does not take place if the Pooling connection string attribute is set to false. Self-tuning is also not supported inside .NET stored procedures.

---

The statement cache size (StatementCacheSize) is tuned automatically by monitoring the statements that are executed by the application. The following sections discuss self-tuning in applications:

- Self-Tuning Statement Caching
- Enabling or Disabling Self-Tuning for Applications
- Tracing Optimization Changes

Self-Tuning Statement Caching

Statement caching helps improve performance by eliminating the need to re-parse each SQL or PL/SQL statement before execution.

If self-tuning is enabled for an application, then ODP.NET continuously monitors application behavior in order to determine the optimum value for the statement cache size. Any statement cache size value specified in the connection string, configuration file, or registry is ignored.

When the application first initializes, it uses the default value of statement cache size. As the application executes statements, ODP.NET collects statistics that are used to self-tune the value of statement cache size. Self-tuning of statement cache size results in increased performance.

---

**Note:** To take full advantage of statement caching, you should not dynamically generate statements, with different inline values, for every statement execution. Instead, use parameterized commands to minimize the number of unique statements that need to be executed and cached. This is because only one statement needs to be cached for every unique command text, regardless of the parameter values and the number of times that the statement is executed.

---

The maximum number of statements that can be cached per connection is determined by the MaxStatementCacheSize configuration attribute. The
MaxStatementCacheSize value can be specified in the Windows registry or XML configuration file.

The MaxStatementCacheSize setting is useful in limiting the number of cached statements, as well as the number of open cursors. This is because a cached statement equates to a cursor being opened on the server. For this reason, you should not set MaxStatementCacheSize to a value that is greater than the database MAX_OPEN_CURSORS setting.

The following Windows registry key is used to configure the MaxStatementCacheSize configuration attribute:

HKLM\Software\Oracle\ODP.NET\version\MaxStatementCacheSize

The MaxStatementCacheSize key is of type REG_SZ. It can be set to an integer value between 0 and System.Int32.MaxValue.

The following example sets the MaxStatementCacheSize property in an ADO.NET 2.0, or above, configuration file:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="MaxStatementCacheSize" value="300"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

If self-tuning is disabled for an application, then the value of statement cache size is determined by the settings in the connection string, configuration file, or the registry. If statement cache size is not specified in any of these sources, then the default value of statement cache size is set to 0. To have ODP.NET configured with the same default settings as previous releases of ODP.NET, disable self-tuning and set the StatementCacheSize value to 10.

**See Also:** "Statement Caching” on page 3-56

**Enabling or Disabling Self-Tuning for Applications**

Self-tuning for ODP.NET applications is enabled by default. An application can enable or disable self-tuning using one of the following methods:

- **Self-Tuning Connection String Attribute**
  
  An application can modify the Self Tuning connection string attribute to enable or disable self-tuning for a particular connection pool. The default value for Self Tuning is true.

- **Windows Registry**
  
  An application can enable or disable self-tuning for a particular version of ODP.NET by modifying the following registry entry:

  HKLM\Software\Oracle\ODP.NET\version\SelfTuning

  The SelfTuning key is of type REG_SZ. It can be set to either 1 (enabled) or 0 (disabled).

- **Configuration File**
An ODP.NET application can modify the application configuration file (app.config) or Web configuration file (web.config) to enable or disable self-tuning.

The following example shows how to enable self-tuning in an ADO.NET 2.0 application configuration file:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <add name="SelfTuning" value="1"/>
    </settings>
  </oracle.dataaccess.client>
</configuration>
```

**Note:** If the optimal statement cache size is known for an application, then you can disable self-tuning and set StatementCacheSize to its optimum value in the registry, configuration file, or the application. If self-tuning is disabled and StatementCacheSize is not set at all, then the default value of 0 is used for StatementCacheSize.

**Tracing Optimization Changes**

Applications can trace optimization changes made by self-tuning. All changes to StatementCacheSize are traced. Errors, if any, are also traced.

The TraceLevel used for tracing self-tuning is 64.

**See Also:** Table 2–1, "Configuration Attributes" for details on TraceLevel values

### ODP.NET Types Overview

ODP.NET types represent Oracle native data types and PL/SQL data 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 and PL/SQL data types in a .NET application than .NET types. For example, an OracleDecimal structure holds up to 38 digits of precision, while a .NET Decimal only holds up to 28.

Table 3–11 lists data types supported by ODP.NET and their corresponding ODP.NET types: data types in the first column refer to both Oracle native data types and PL/SQL data types of that name. Those data types that exist only in PL/SQL are indicated by (PL/SQL only) after the data type name. The entries for the PL/SQL data types also represent the subtypes of the data types, if any. The third column lists the .NET Framework data type that corresponds to the Value property of each ODP.NET type.

<table>
<thead>
<tr>
<th>Oracle Native Data Type or PL/SQL Data Type</th>
<th>ODP.NET Type</th>
<th>.NET Framework Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFILE</td>
<td>OracleBFile class</td>
<td>System.Byte[]</td>
</tr>
<tr>
<td>BINARY_DOUBLE</td>
<td>OracleDecimal structure</td>
<td>System.Decimal</td>
</tr>
</tbody>
</table>
**Table 3-11 (Cont.) Value Property Type of ODP.NET Type**

<table>
<thead>
<tr>
<th>Oracle Native Data Type or PL/SQL Data Type</th>
<th>ODP.NET Type</th>
<th>.NET Framework Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINARY_FLOAT</td>
<td>OracleDecimal structure</td>
<td>System.Decimal</td>
</tr>
<tr>
<td>BINARY_INTEGER (PL/SQL only)</td>
<td>OracleDecimal structure</td>
<td>System.Decimal</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>
<tr>
<td>NCHAR</td>
<td>OracleString structure</td>
<td>System.String</td>
</tr>
<tr>
<td>NCLOB</td>
<td>OracleClob class</td>
<td>System.String</td>
</tr>
<tr>
<td>NUMBER</td>
<td>OracleDecimal structure</td>
<td>System.Decimal</td>
</tr>
<tr>
<td>NVARCHAR2</td>
<td>OracleString structure</td>
<td>System.String</td>
</tr>
<tr>
<td>PLS_INTEGER (PL/SQL only)</td>
<td>OracleDecimal structure</td>
<td>System.Decimal</td>
</tr>
<tr>
<td>RAW</td>
<td>OracleBinary structure</td>
<td>System.Byte[]</td>
</tr>
<tr>
<td>REF</td>
<td>OracleRef class</td>
<td>System.String</td>
</tr>
<tr>
<td>REF CURSOR (PL/SQL only)</td>
<td>OracleRefCursor class</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>ROWID</td>
<td>OracleString structure</td>
<td>System.String</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>OracleTimeStamp structure</td>
<td>System.DateTime</td>
</tr>
<tr>
<td>TIMESTAMP WITH LOCAL TIME ZONE</td>
<td>OracleTimeStampLTZ structure</td>
<td>System.DateTime</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>OracleTimeStampTZ structure</td>
<td>System.DateTime</td>
</tr>
<tr>
<td>UROWID</td>
<td>OracleString structure</td>
<td>System.String</td>
</tr>
<tr>
<td>VARCHAR2</td>
<td>OracleString structure</td>
<td>System.String</td>
</tr>
<tr>
<td>XMLType</td>
<td>OracleXmlType class</td>
<td>System.String</td>
</tr>
</tbody>
</table>
Obtaining Data from an OracleDataReader Object

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` object:

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

Typed OracleDataReader Accessors

The `OracleDataReader` class provides two types of typed accessors:

- .NET Type Accessors
- ODP.NET Type Accessors

.NET Type Accessors

Table 3–12 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 data types depend on the version of the database; therefore, some data types are not available in earlier versions of Oracle Database.

See Also:

- "OracleDataAdapter Class" on page 5-116
- "OracleDataReader Class" on page 5-152

<table>
<thead>
<tr>
<th>Oracle Native Data Type</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 DAY TO SECOND</td>
<td>System.TimeSpan</td>
<td>GetTimeSpan</td>
</tr>
<tr>
<td>INTERVAL YEAR TO MONTH</td>
<td>System.Int64</td>
<td>GetInt64</td>
</tr>
</tbody>
</table>
Certain methods and properties of the `OracleDataReader` object 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:

<table>
<thead>
<tr>
<th>Oracle Native Data Type</th>
<th>.NET Type</th>
<th>Typed Accessor</th>
</tr>
</thead>
<tbody>
<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>REF</td>
<td>System.String</td>
<td>GetString</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>
<tr>
<td>VARCHAR2</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Char[]</td>
<td>GetChars</td>
</tr>
<tr>
<td>XMLType</td>
<td>System.String</td>
<td>GetString</td>
</tr>
<tr>
<td></td>
<td>System.Xml.XmlReader</td>
<td>GetXmlReader</td>
</tr>
</tbody>
</table>
Obtaining Data from an OracleDataReader Object

- 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 type. If the value in the column cannot be represented as 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 type. If a value in the column cannot be represented as a System.Decimal type, then an exception is raised.

The Fill method of the OracleDataAdapter class uses the OracleDataReader object 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.

See Also:
- "OracleDataReader Class" on page 5-152
- "OracleDataAdapter Class" on page 5-116
- "Item" on page 5-164
- "GetFieldTo" on page 5-180
- "GetValues" on page 5-212
- "GetValue" on page 5-211

**ODP.NET Type Accessors**

ODP.NET exposes provider-specific types that natively represent the data types in the database. In some cases, these ODP.NET types provide better performance and functioning than the corresponding .NET types. The ODP.NET types can be obtained from the OracleDataReader object by calling their respective typed accessor.

See Also: "ODP.NET Types Overview" on page 3-60 for a list of all ODP.NET types

Table 3–13 lists the valid type accessors that ODP.NET uses to obtain ODP.NET types for an Oracle native type.

**Table 3–13 ODP.NET Type Accessors**

<table>
<thead>
<tr>
<th>Oracle Native Data 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>
</tbody>
</table>
Obtaining Data from an OracleDataReader Object

Obtaining LONG and LONG RAW Data

ODP.NET fetches and caches rows from the database during the Read method invocations on the OracleDataReader object. The amount of LONG and LONG RAW column data that is retrieved from this operation is determined by InitialLONGFetchSize. The different behaviors observed when InitialLONGFetchSize is set to 0, greater than 0, and -1 are explained in the following sections.
Obtaining Data from an OracleDataReader Object

---

**Note:** ODP.NET does not support the `CommandBehavior.SequentialAccess` enumeration value. Therefore, LONG and LONG RAW data can be fetched randomly.

---

### Setting InitialLONGFetchSize to Zero or a Value Greater than Zero

The specified amount of `InitialLONGFetchSize` characters or bytes for LONG or LONG RAW column data is retrieved into the cache during the `Read` method invocations on the `OracleDataReader` object.

By default, `InitialLONGFetchSize` is set to 0. In this case, ODP.NET does not fetch any LONG or LONG RAW column data during the `Read` method invocations on the `OracleDataReader` object. The LONG or LONG RAW data is fetched when the typed accessor method is explicitly invoked for the LONG or LONG RAW column, which incurs a database round-trip because no data is cached.

If `InitialLONGFetchSize` is set to a value greater than 0, that amount of specified data is cached by ODP.NET during the `Read` method invocations on the `OracleDataReader` object. If the application requests an amount of data less than or equal to the `InitialLONGFetchSize` through the typed accessor methods, no database round-trip is incurred. However, an additional database round-trip is required to fetch data beyond `InitialLONGFetchSize`.

To obtain data beyond the `InitialLONGFetchSize` characters or bytes, 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)

To be able to fetch the entire LONG or LONG RAW data without having a primary key column, a ROWID, or unique columns in the select list, set the size of the `InitialLONGFetchSize` property on the `OracleCommand` object to equal or greater than the number of characters or bytes needed to be retrieved.

The LONG or LONG RAW data is returned when the appropriate typed accessor method (GetChars, GetOracleString, or GetString for LONG or GetOracleBinary or GetBytes for LONG RAW) is called on the `OracleDataReader` object.

### Setting InitialLONGFetchSize to -1

By setting `InitialLONGFetchSize` to -1, it is possible to fetch the entire LONG or LONG RAW data from the database for a select query, without requiring a primary key, ROWID, or unique column in the select list.

When `InitialLONGFetchSize` is set to -1, the entire LONG or LONG RAW data is retrieved and cached during `Read` method invocations on the `OracleDataReader` object. Calls to `GetString`, `GetOracleString`, `GetChars`, `GetBytes`, or `GetOracleBinary` in the `OracleDataReader` return the entire column data.

### Obtaining LOB Data

ODP.NET fetches and caches rows from the database during the `Read` method invocations on the `OracleDataReader` object. The amount of LOB column data that is retrieved from this operation is determined by `InitialLOBFetchSize`.

---
The following sections explain the different behaviors observed when `InitialLOBFetchSize` is set to 0, greater than 0, and -1.

**Setting InitialLOBFetchSize to Zero**

By default, when the `InitialLOBFetchSize` property is 0, the `GetOracleBlob` and `GetOracleClob` methods can be invoked on the `OracleDataReader` object to obtain `OracleBlob` and `OracleClob` objects.

The following is a complete list of typed accessor methods that an application can call for the `CLOB` and `BLOB` columns, if `InitialLOBFetchSize` is set to 0:

- **Methods callable for BLOB column**
  - `GetBytes`
  - `GetValue`
  - `GetValues`
  - `GetOracleBinary`
  - `GetOracleBlob`
  - `GetOracleBlobForUpdate`
  - `GetOracleValue`
  - `GetOracleValues`

- **Methods callable for CLOB column**
  - `GetChars`
  - `GetString`
  - `GetValue`
  - `GetValues`
  - `GetOracleString`
  - `GetOracleClob`
  - `GetOracleClobForUpdate`
  - `GetOracleValue`
  - `GetOracleValues`

**Setting InitialLOBFetchSize to a Value Greater than Zero**

If `InitialLOBFetchSize` is set to a value greater than 0, ODP.NET caches LOB data up to `InitialLOBFetchSize` characters or bytes for each LOB selected during the `Read` method invocations on the `OracleDataReader` object.

This section discusses the ways to fetch beyond the `InitialLOBFetchSize` characters or bytes that are cached. The functionality has changed from Oracle Database 10g release 2 (10.2) and later.

**Obtaining Additional Data Prior to Oracle Database 10g Release 2 (10.2)** With releases prior to Oracle Database 10g release 2 (10.2), obtaining data beyond `InitialLOBFetchSize` characters or bytes requires one of the following in the query select list:

- Primary key
- `ROWID`
 Obtaining Data from an OracleDataReader Object

- 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 LOB data is fetched from the database when the appropriate typed accessor method is called on the OracleDataReader object.

To be able to fetch the entire LOB data without having a primary key column, a ROWID, or unique columns in the select list, set the size of the InitialLOBFetchSize property on the OracleCommand object to equal or greater than the number of characters or bytes needed to be retrieved.

When the InitialLOBFetchSize property is set to a nonzero value, the GetOracleBlob, GetOracleClob, GetOracleBlobForUpdate, and GetOracleClobForUpdate typed accessor methods are disabled.

Obtaining Additional Data From Oracle Database 10g Release 2 (10.2) and Later Starting with Oracle Database 10g release 2 (10.2), the entire LOB data is returned when a typed accessor is invoked, regardless of the value set to the InitialLOBFetchSize property. Primary key, ROWID, or unique columns are not required to be in the query select list to obtain data beyond the specified InitialLOBFetchSize.

The GetOracleBlob, GetOracleClob, GetOracleBlobForUpdate, and GetOracleClobForUpdate methods can now be invoked even if InitialLOBFetchSize is greater than 0, starting with Oracle Database 10g release 2.

The following is a complete list of typed accessor methods that an application can call for the CLOB and BLOB columns if InitialLOBFetchSize is set to a value greater than 0:

- Methods callable for BLOB column
  - GetBytes
  - GetValue
  - GetValues
  - GetOracleBinary
  - GetOracleBlob
  - GetOracleBlobForUpdate
  - GetOracleValue
  - GetOracleValues

- Methods callable for CLOB column
  - GetChars
  - GetString
  - GetValue
  - GetValues
  - GetOracleString
  - GetOracleClob
  - GetOracleClobForUpdate
  - GetOracleValue
- GetOracleValues

**Setting InitialLOBFetchSize to -1**

By setting `InitialLOBFetchSize` to -1, it is possible to fetch the entire LOB data from the database for a select query, without requiring a primary key, `ROWID`, or unique column in the select list. When `InitialLOBFetchSize` is set to -1, the entire LOB column data is fetched and cached during the Read method invocations on the `OracleDataReader` object. Calls to `GetString`, `GetOracleString`, `GetChars`, `GetBytes`, or `GetOracleBinary` in the `OracleDataReader` allow retrieving all data.

**Methods Supported for InitialLOBFetchSize of -1**

This section lists supported and not supported methods for the CLOB and BLOB data types when the `InitialLOBFetchSize` property is set to -1.

Table 3–14 lists supported and not supported methods for the CLOB data types.

<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></td>
</tr>
<tr>
<td>GetValues</td>
<td></td>
</tr>
<tr>
<td>GetOracleString</td>
<td></td>
</tr>
<tr>
<td>GetOracleValue</td>
<td></td>
</tr>
<tr>
<td>GetOracleValues</td>
<td></td>
</tr>
</tbody>
</table>

Table 3–15 lists supported and not supported methods for the BLOB data types.

<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></td>
</tr>
<tr>
<td>GetOracleBinary</td>
<td></td>
</tr>
<tr>
<td>GetOracleValue</td>
<td></td>
</tr>
<tr>
<td>GetOracleValues</td>
<td></td>
</tr>
</tbody>
</table>

**Performance Considerations Related to the InitialLOBFetchSize Property**

This section discusses the advantages and disadvantages of the various `InitialLOBFetchSize` property settings in different situations. It also discusses ways to enhance performance, depending on which database release you are using.

**Prior to Oracle Database 10g Release 2 (10.2)**

Setting the `InitialLOBFetchSize` property to a nonzero value can improve performance in certain cases. Using the `InitialLOBFetchSize` property 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
Obtaining Data from an OracleDataReader Object

OracleDataReader object and the size of the LOB column data is not very large. The InitialLOBFetchSize property is particularly useful in cases where the size of the LOB column data returned by the query is approximately the same for all the rows.

It is generally recommended that the InitialLOBFetchSize property 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 the InitialLOBFetchSize property to 1 KB.

See Also:
- "LOB Support" on page 3-83
- "InitialLOBFetchSize" on page 5-18
- "InitialLONGFetchSize" on page 5-20

Oracle Database 10g Release 2 (10.2) and Later  An application does not have to choose between performance and OracleBlob and OracleClob functionality. Setting the InitialLOBFetchSize property results in a performance boost and still gives the flexibility to use the OracleBlob and OracleClob objects.

If the size of the LOB data is unknown or if the LOB data size varies irregularly, then it is better to leave the InitialLOBFetchSize property to its default value of 0. This still gives better performance in most cases.

Setting the InitialLOBFetchSize property to a size equal to or greater than the LOB data size for most rows improves performance. It is generally recommended that the InitialLOBFetchSize property 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 the InitialLOBFetchSize property to 1 KB.

See Also:
- "LOB Support" on page 3-83
- "InitialLOBFetchSize" on page 5-18
- "InitialLONGFetchSize" on page 5-20

Controlling the Number of Rows Fetched in One Database 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 database round-trip.

The FetchSize property can be set on the OracleCommand, OracleDataReader, or OracleRefCursor object, depending on the situation. It controls the fetch size for filling a DataSet or DataTable using an OracleDataAdapter.

If the FetchSize property is set on the OracleCommand object, then the newly created OracleDataReader object inherits the FetchSize property of the OracleCommand object. This inherited FetchSize value 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 property. All subsequent fetches from the
Obtaining Data from an OracleDataReader Object

Features of Oracle Data Provider for .NET

3-71

database use the same cache allocated for that OracleDataReader object. Therefore, changing the FetchSize value 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 database round-trip for better performance.

For example, if a query returns 100 rows and each row takes 1024 bytes, then setting the FetchSize property to 10240 takes just one database round-trip to fetch 100 rows. For the same query, if the FetchSize property is set to 10240, it takes 10 database 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 because it fetches only 10 rows, not 100 rows. When the next 10 rows are fetched, then the memory allocated for rows 1-10 is reused for rows 11-20.

The larger the FetchSize, the more system memory is used. Developers should not set large fetch sizes if their client systems have limited memory resources.

Using the RowSize Property
The RowSize property of the OracleCommand or OracleRefCursor 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 property by setting it to the result of multiplying the RowSize value times the number of rows to fetch for each database round-trip.

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

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

Setting FetchSize Value in the Registry The HKLM\Software\Oracle\ODP.NET\version\FetchSize registry entry can be set to specify the default result set fetch size (in bytes) for all applications that use that particular version of ODP.NET or the FetchSize attribute in the application configuration or web.config file can specify the default value for a given application. By default, the fetch size is 131072 bytes. This value can be overridden programmatically by having the applications set the FetchSize property on either the OracleCommand or the OracleDataReader at run time.

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

### Setting FetchSize Value at Run Time
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` object, 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` object to the product of the `RowSize` property value multiplied by the number of rows the application wishes to fetch for each database round-trip. In this scenario, the `FetchSize` property is set by accessing the `RowSize` property at run time.

### PL/SQL REF CURSOR and OracleRefCursor
The `REF CURSOR` is a data type in the Oracle PL/SQL language. It represents a cursor or a result set in Oracle Database. The `OracleRefCursor` object is a corresponding ODP.NET type for the `REF CURSOR` type.

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

- Obtaining an `OracleRefCursor` Object
- Obtaining a REF CURSOR Data Type
- Populating an `OracleDataReader` from a REF CURSOR
- Populating the DataSet from a REF CURSOR
- Populating an `OracleRefCursor` from a REF CURSOR
- Updating a DataSet Obtained from a REF CURSOR
- Behavior of ExecuteScalar Method for REF CURSOR
- Passing a REF CURSOR to a Stored Procedure

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

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

### Obtaining a REF CURSOR Data Type
A REF CURSOR data type can be obtained as an `OracleDataReader`, `DataSet`, or `OracleRefCursor` object. If the REF CURSOR data type is obtained as an `OracleRefCursor` object, it can be used to create an `OracleDataReader` object or populate a `DataSet` from it. When accessing a REF CURSOR data type, always bind it as an `OracleDbType` and `RefCursor` parameter.
Populating an **OracleDataReader** from a REF CURSOR

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

If there are multiple output REF CURSOR parameters, use the **NextResult** method of the **OracleDataReader** object to access the next REF CURSOR data type. The **OracleDataReader** NextResult method provides sequential access to the REF CURSOR data types; only one REF CURSOR data type can be accessed at a given time.

The order in which **OracleDataReader** objects are created for the corresponding REF CURSOR data types depends on the order in which the parameters are bound. If a PL/SQL stored function returns a REF CURSOR data type, then it becomes the first **OracleDataReader** object and all the output REF CURSOR data types 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** property of the **OracleDataAdapter** class 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 data type.

If the command execution returns multiple REF CURSOR data types, the **DataSet** is populated with multiple **DataTable** objects.

With Oracle Data Provider for .NET release 11.1.0.6.20, the extended property, **REFCursorName**, has been introduced on the **DataTable**, to identify the REF CURSOR that populates the **DataTable**.

This property is particularly useful when a **DataSet** is being populated with more than one REF CURSOR, one or more of which is NULL. For example, if a **DataSet** is populated by executing a stored procedure that returns three REF CURSORs and the second REF CURSOR is NULL, the **REFCursorName** property value for the first **DataTable** is **REFCursor** and for the second **DataTable**, **REFCursor2**. No **DataTable** is populated for the NULL REF CURSOR.

Populating an **OracleRefCursor** from a REF CURSOR

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

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

To populate a **DataSet** with an **OracleRefCursor** object, the application can invoke a **Fill** method of the **OracleDataAdapter** class that takes an **OracleRefCursor** object. Similar to the **OracleDataReader** object, an **OracleRefCursor** object is forward-only. Therefore, once a row is read from an **OracleRefCursor** object, that same row cannot be obtained again from it unless it is populated again from a query.

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

**Updating a DataSet Obtained from a REF CURSOR**

REF CURSOR types cannot be updated. However, data that is retrieved into a DataSet can be updated. Therefore, the OracleDataAdapter class requires a custom SQL statement to flush any REF CURSOR data updates to the database.

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

**Behavior of ExecuteScalar Method for REF CURSOR**

The `ExecuteScalar` method returns the value of the first column of the first row of the REF CURSOR if it is one of the following:

- A return value of a stored function execution
- The first bind parameter of a stored procedure execution

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

**Passing a REF CURSOR to a Stored Procedure**

An application can retrieve a REF CURSOR type from a PL/SQL stored procedure or function and pass it to another stored procedure or function. This feature is useful in scenarios where a stored procedure or a function returns a REF CURSOR type to the .NET application, and based on the application logic, the application passes this REF CURSOR to another stored procedure for processing. Note that if you retrieve the data from a REF CURSOR type in the .NET application, you cannot pass it back to another stored procedure.

The following example demonstrate passing a REF CURSOR:

```sql
/*
connect scott/tiger@oracle
create table test (col1 number);
insert into test(col1) values (1);
commit;

create or replace package testPkg as type empCur is REF Cursor;
end testPkg;
/

create or replace procedure testSP(param1 IN testPkg.empCur, param2 OUT NUMBER)
as
begin
FETCH param1 into param2;
end;
/
*/

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

See Also: *Oracle Database Application Developer’s Guide - Large Objects* for more information
using System.Data;

class InRefCursorParameterSample
{
    static void Main()
    {
        OracleConnection conn = new OracleConnection
        {"User Id=scott; Password=tiger; Data Source=oracle"};

        conn.Open(); // Open the connection to the database

        // Command text for getting the REF Cursor as OUT parameter
        String cmdTxt1 = "begin open :1 for select col1 from test; end;";

        // Command text to pass the REF Cursor as IN parameter
        String cmdTxt2 = "begin testSP (:1, :2); end;";

        // Create the command object for executing cmdTxt1 and cmdTxt2
        OracleCommand cmd = new OracleCommand(cmdTxt1, conn);

        // Bind the Ref cursor to the PL/SQL stored procedure
        OracleParameter outRefPrm = cmd.Parameters.Add("outRefPrm",
            OracleDbType.RefCursor, DBNull.Value, ParameterDirection.Output);
        cmd.ExecuteNonQuery(); // Execute the anonymous PL/SQL block

        // Reset the command object to execute another anonymous PL/SQL block
        cmd.Parameters.Clear();
        cmd.CommandText = cmdTxt2;

        // REF Cursor obtained from previous execution is passed to this
        // procedure as IN parameter
        OracleParameter inRefPrm = cmd.Parameters.Add("inRefPrm",
            OracleDbType.RefCursor, outRefPrm.Value, ParameterDirection.Input);

        // Bind another Number parameter to get the REF Cursor column value
        OracleParameter outNumPrm = cmd.Parameters.Add("outNumPrm",
            OracleDbType.Int32, DBNull.Value, ParameterDirection.Output);

        cmd.ExecuteNonQuery(); // Execute the stored procedure

        // Display the out parameter value
        Console.WriteLine("out parameter is: " + outNumPrm.Value.ToString());
    }
}

Implicit REF CURSOR Binding

ODP.NET 11g Release 2 (11.2.0.3.0), and higher, enables applications to run stored procedures with REF CURSOR parameters without using explicit binding for these parameters in the .NET code.

For a read-only result set, such as a REF CURSOR using OracleDataReader, REF CURSOR schema information is retrieved automatically.

For some scenarios, such as when updateable REF CURSORs or Entity Framework is used, developers need to define the REF CURSOR schema information so that the application can bind the implicit REF CURSOR. Entity Framework applications use implicit REF CURSOR binding to instantiate complex types from REF CURSOR data.
Applications must specify REF CURSOR bind and metadata information in the app.config, web.config, or machine.config .NET configuration file.

The attributes supplied in the .NET configuration file are also used when the application requests for schema information from the OracleDataReader object that represents a REF CURSOR. This means that for REF CURSORS that are created using a SELECT from a single table, the application can update that table through the use of OracleDataAdapter and OracleCommandBuilder.

When using the Entity Framework, function imports can return an implicitly-bound REF CURSOR. The REF CURSOR can be returned as a collection of complex types or entity types. To return a complex type collection, the .NET configuration file needs to define the REF CURSOR bind and metadata information. To return an entity type collection, only the bind information needs to be defined in the .NET configuration file.

This section contains the following topics:

- Specifying REF CURSOR Bind and Metadata Information in the .NET Configuration File
- Sample Configuration File and Application
- Usage Considerations

### Specifying REF CURSOR Bind and Metadata Information in the .NET Configuration File

Specify the REF CURSOR information in the oracle.dataaccess.client configuration section of the .NET configuration file. Use an `<add>` element for each piece of information. The `add` element uses name-value attributes to specify REF CURSOR information. Use the following format to specify bind information:

```xml
<add
    name="SchemaName.PackageName.StoredProcedureName.RefCursor.RefCursorParameterPositionOrName"
    value="implicitRefCursor bindinfo='mode=InputOutput|Output|ReturnValue'" />
```

Use the following format to specify metadata information:

```xml
<add
    name="SchemaName.PackageName.StoredProcedureName.RefCursorMetaData.RefCursorParameterPositionOrName.Column.ColumnOrdinal"
    value="implicitRefCursor metadata=AttributesList" />
```

Each REF CURSOR column needs to have an `add` element defined for it. For example, if you have a REF CURSOR returning five columns, then you need to define five `add` elements in the config file.

Each `add` element contains the name and value attributes. The `value` attribute must begin with the word `implicitRefCursor` followed by the `bindinfo` or `metadata` attribute for specifying bind or metadata information.

The `bindinfo` information is used by ODP.NET for binding REF CURSOR parameters. The `metadata` information is used by ODP.NET to associate the schema information with the appropriate REF CURSOR. The metadata comprises of an attributes list that includes parameters together with their values.

The `SchemaName`, `PackageName`, and `.StoredProcedureName` are case-sensitive. In order to run a stored procedure with implicit REF CURSOR binding, the `SchemaName.PackagENAME.StoredProcedureName` portion of the name attribute
must exactly match the name specified in the data dictionary for that stored procedure.

**Note:** If the application uses implicit REF CURSOR binding feature outside of Entity Framework, then the .NET configuration file and OracleCommand CommandText do not require the schema name concatenated before the stored procedure name.

If any schema, package, or stored procedure name in the database contains lowercase characters, then it must be enclosed within double quotation marks ("”) in the config file to preserve the case. Double quotation marks are used within the name attribute by using &quot; when needed. For example, if the schema name is HrSchema, the package name is HrPackage, and the stored procedure name is HrStoredProcedure in the database, the config file should use the following:

```xml
<add name="&quot;HrSchema&quot;.&quot;HrPackage&quot;.&quot;HrStoredProcedure&quot;.RefCursorMetaData . . . />
```

By default, Oracle Database stores these names as uppercase characters. ODP.NET assumes default behavior, and converts all names to uppercase characters unless you explicitly preserve the case by using double quotation marks.

**Note:** The SchemaName, PackageName, StoredProcedureName, or ParameterName cannot contain a period (".") in the name. For example, P.0 is an unacceptable parameter name.

Depending on whether the application uses bind-by-name or bind-by-position, the RefCursorParameterPositionOrName portion of the name attribute must be set with the correct parameter position (for bind by position) or parameter name (for bind by name). For functions, the position is 0-based, where the position 0 represents the return value. For procedures, the position is 1-based, as there are no return values for procedures. For example, if a stored procedure accepts five parameters, returning only two REF CURSORs in the third and fifth parameter positions, then the .NET config REF CURSOR bind information should contain one entry for position 3 and one entry for position 5.

If bind-by-name is used, the attribute name is used to identify the REF CURSOR parameter. The name should use the same name and case as the one specified in the data dictionary for that stored procedure.

For bindinfo, the mode specifies the parameter direction of the parameter. The mode must be either InputOutput, Output, or ReturnValue.

**Note:** Implicit REF CURSOR binding for an input REF CURSOR parameter is not supported.

An exception is thrown at runtime if the .NET configuration file contains an entry for a REF CURSOR whose mode is set to Input.

For metadata, The AttributesList contains the list of parameters. Table 3–16 describes the parameters that can be included in the AttributesList.
Example 3–2 shows a sample add element that uses bindinfo. Here, the schema name is SCOTT and the stored procedure name is TESTPROC. The parameter name is parameter1. The mode is output.

Example 3–2  Using the add Element with bindinfo

```xml
<add name="SCOTT.TESTPROC.RefCursor.parameter1" value="implicitRefCursor
bindinfo='mode=Output'" />
```

Example 3–3 shows a sample add element that uses metadata.

Example 3–3  Using the add Element with metadata

```xml
<add name="scott.TestProc.RefCursorMetaData.parameter1.Column.0" value="implicitRefCursor metadata='ColumnName=EMPNO;BaseColumnName=EMPNO;
BaseSchemaName=SCOTT;BaseTableName=EMP;NativeDataType=number;
ProviderType=Int32;DataType=System.Int32;ColumnSize=4;AllowDBNull=false;
IsKey=true'" />
```

Table 3–16  Allowed Parameters in Attributes List

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required/Optional for Entity Framework</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnName</td>
<td>System.String</td>
<td>Required</td>
<td>The name of the column.</td>
</tr>
<tr>
<td>ProviderType</td>
<td>Oracle.DataAccess.Client.OraclDbType</td>
<td>Required</td>
<td>The database column type (OracleDbType) of the column</td>
</tr>
<tr>
<td>NativeDataType</td>
<td>System.String</td>
<td>Required</td>
<td>The Oracle type. For example, NCLOB.</td>
</tr>
<tr>
<td>BaseColumnName</td>
<td>System.String</td>
<td>Optional</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>Optional</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>Optional</td>
<td>The name of the table or view in the database that contains the column.</td>
</tr>
<tr>
<td>ColumnSize</td>
<td>System.Int64</td>
<td>Optional</td>
<td>The maximum possible length of a value in the column.</td>
</tr>
<tr>
<td>NumericPrecision</td>
<td>System.Int16</td>
<td>Optional</td>
<td>The maximum precision of the column, if the column is a numeric data type.</td>
</tr>
<tr>
<td>NumericScale</td>
<td>System.Int16</td>
<td>Optional</td>
<td>The maximum scale of the column, if the column is a numeric data type.</td>
</tr>
<tr>
<td>IsUnique</td>
<td>System.Boolean</td>
<td>Optional</td>
<td>Indicates whether or not the column is unique.</td>
</tr>
<tr>
<td>IsKey</td>
<td>System.Boolean</td>
<td>Optional</td>
<td>Indicates whether or not the column is a key column. For a table to be updated with the REF CURSOR information, at least one of the columns in the REF CURSOR metadata should have this value set to true</td>
</tr>
</tbody>
</table>
Some of the attributes, listed in Table 3–16, automatically have their values set using the result set’s metadata. Developers can override these default values by setting a value explicitly.

You may have to explicitly define some attributes listed as optional for certain operations. For example, updateable REF CURSOR requires the developer to define key information.

**Sample Configuration File and Application**

This section builds a sample application to illustrate implicit REF CURSOR binding. It contains the following topics:

- Sample Stored Procedure and Function
- Sample Application Configuration File
- Sample Application That Uses the Configuration File

**Sample Stored Procedure and Function**

CREATE OR REPLACE FUNCTION GETEMP (IsRowID System.Boolean Optional true if the column is a ROWID, otherwise false.

DataType System.Runtime Type Optional Maps to the common language runtime type.

AllowDBNull System.Boolean Optional true if null values are allowed, otherwise false.

IsAliased System.Boolean Optional true if the column is an alias; otherwise false.

IsByteSemantic System.Boolean Optional IsByteSemantic is:

- true if the ColumnSize value uses bytes semantics
- false if ColumnSize uses character semantics

IsExpression System.Boolean Optional true if the column is an expression, else false.

IsHidden System.Boolean Optional true if the column is hidden, else false.

IsReadOnly System.Boolean Optional true if the column is read-only, else false.

IsLong System.Boolean Optional true if the column is of LONG, LONG RAW, BLOB, CLOB, or BFILE type, else false.

UdtTypeName System.String Optional The type name of the UDT.

ProviderDBType System.Data.Db Type Optional System.Data.DbType

ObjectName System.String Optional Represents the name of the object.
EMPID IN NUMBER; return sys_refcursor is
emp sys_refcursor is
BEGIN
  OPEN emp FOR SELECT empno, ename FROM emp where empno = EMPID;
  return emp;
END;
/

CREATE OR REPLACE PROCEDURE "GetEmpAndDept" (EMPS OUT sys_refcursor,
  DEPTS OUT sys_refcursor) AS
BEGIN
  OPEN EMPS for SELECT empno, ename from emp;
  OPEN DEPTS for SELECT deptno, dname from dept;
END;
/

Sample Application Configuration File
<configuration>
<oracle.dataaccess.client>

<!-- The following is for SCOTT.GETEMP -->
<add name="SCOTT.GETEMP.RefCursor.0" value="implicitRefCursor bindinfo='mode=ReturnValue'" />

<!-- The following is for SCOTT.GETEMP's REF CURSOR metadata -->
<add name="SCOTT.GETEMP.RefCursorMetaData.0.Column.0" value="implicitRefCursor metadata='ColumnName=EMPNO;BaseColumnName=EMPNO;BaseSchemaName=SCOTT;BaseTableName=EMP;NativeDataType=number;ProviderType=Int32;ProviderDBType=Int32;DataType=System.Int32;ColumnSize=4;NumericPrecision=10;NumericScale=3;AllowDBNull=false;IsKey=true'" />

<add name="SCOTT.GETEMP.RefCursorMetaData.0.Column.1" value="implicitRefCursor metadata='ColumnName=ENAME;BaseColumnName=ENAME;BaseSchemaName=SCOTT;BaseTableName=EMP;NativeDataType=varchar2;ProviderType=Varchar2;ProviderDBType=String;DataType=System.String;ColumnSize=10;AllowDBNull=true'" />

<!-- The following is for 'SCOTT'."GetEmpAndDept" -->
<add name="SCOTT."GetEmpAndDept".RefCursor.EMPS value="implicitRefCursor bindinfo='mode=Output'" />

<!-- The following is for SCOTT.GETEMP's EMPS REF CURSOR metadata -->
<add name="SCOTT."GetEmpAndDept".RefCursorMetaData.EMPS.Column.0" value="implicitRefCursor metadata='ColumnName=EMPNO;BaseColumnName=EMPNO;BaseSchemaName=SCOTT;BaseTableName=EMP;NativeDataType=number;ProviderType=Int32;ProviderDBType=Int32;DataType=System.Int32;ColumnSize=4;NumericPrecision=10;NumericScale=3;AllowDBNull=false;IsKey=true'" />

<add name="SCOTT."GetEmpAndDept".RefCursorMetaData.EMPS.Column.1" value="implicitRefCursor metadata='ColumnName=ENAME;BaseColumnName=ENAME;BaseSchemaName=SCOTT;BaseTableName=EMP;NativeDataType=varchar2;ProviderType=Varchar2;ProviderDBType=String;DataType=System.String;ColumnSize=10;AllowDBNull=true'" />
</oracle.dataaccess.client>
</configuration>
Implicit REF CURSOR Binding

Features of Oracle Data Provider for .NET

Sample Application That Uses the Configuration File

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

class Program
{
    static void Main(string[] args)
    {
        try
        {
            // Open a connection
            string constr = "User Id=scott;Password=tiger;Data Source=inst1";
            OracleConnection con = new OracleConnection(constr);
            con.Open();

            // Use implicit REF CURSOR binding
            // to execute SCOTT.GETEMP function
            // Use bind by position as configured
            // in app.config for SCOTT.GETEMP
            OracleCommand cmd = con.CreateCommand();
            cmd.CommandText = "SCOTT.GETEMP";
            cmd.CommandType = CommandType.StoredProcedure;
            OracleParameter empid = cmd.Parameters.Add("empid", OracleDbType.Int32, ParameterDirection.Input);
            empid.Value = 7654;

            // Populate the DataSet
            OracleDataAdapter adapter = new OracleDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds);
            Console.WriteLine("Retrieved {0} row from EMP", ds.Tables[0].Rows.Count);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

Features of Oracle Data Provider for .NET 3-81
ds.Tables[0].Rows.Count);

// Use implicit REF CURSOR binding
// to execute 'SCOTT'."GetEmpAndDept" procedure
// Use bind by name as configured
// in app.config for 'SCOTT'."GetEmpAndDept"
cmd = con.CreateCommand();
cmd.CommandText = "'SCOTT'."GetEmpAndDept";";
var CommandType = CommandType.StoredProcedure;
adapter = new OracleDataAdapter(cmd);
adapter.Fill(ds);
Console.WriteLine("Retrieved {0} rows from DEPT",
    ds.Tables[1].Rows.Count);
}
catch (Exception ex)
{
    // Output the message
    Console.WriteLine(ex.Message);
    if (ex.InnerException != null)
    {
        // If any details are available regarding
        // errors in the app.config, print them out
        Console.WriteLine(ex.InnerException.Message);
        if (ex.InnerException.InnerException != null)
        {
            Console.WriteLine(
                ex.InnerException.InnerException.Message);
        }
    }
}

Usage Considerations

This section discusses the following usage considerations when using implicit REF CURSOR:

- **CommandText Property Considerations**
- **Bind Considerations**
- **Overloaded Stored Procedures**
- **Type Initialization Exceptions**
- **Using Stored Functions with Function Import**

**CommandText Property Considerations**

ODP.NET applications should ensure that the stored procedure name and the OracleCommand CommandText match exactly. Let's take a scenario where the stored procedure name in the database is `SCOTT.TESTPROC`. Now, if the CommandText uses `TESTPROC`, ODP.NET will look for entries matching `TESTPROC` only. The current schema name will not be automatically appended to `TESTPROC`. So, the correct CommandText to use in this scenario would be `SCOTT.TESTPROC`.

Also, the CommandText is case-sensitive and must use the same case as the stored procedure name in the database. So if the stored procedure name in the database is `SCOTT.Testproc`, then the CommandText must use `SCOTT.Testproc`.

---

3-82 Oracle Data Provider for .NET Developer's Guide
Bind Considerations
If information about a REF CURSOR parameter has been added to the configuration file, then applications should not try to explicitly bind the REF CURSOR parameter to OracleCommand. ODP.NET automatically binds the REF CURSOR parameter at the appropriate locations based on the information provided in the configuration file. If the application stored procedure also has non-REF CURSOR parameters, then these parameters must still be explicitly bound to OracleCommand.

If the information specified in the configuration file for a stored procedure identifies the REF CURSOR parameter by name, then all the other non-REF CURSOR parameters should also be bound by name. Also the BindByName property for the OracleCommand object should be set to true in this case. Entity Framework always uses BindByName to run stored procedures. Your .NET configuration file parameter names must use the same case that was used when creating the stored procedure in the database.

If the OracleCommand BindByName property is set to false (default), then ODP.NET assumes that the parameters have been bound based on their position, and all parameters have been specified in the correct order. For such cases, the parameters specified in the configuration file are bound in the same order in which they appear in the configuration file.

Overloaded Stored Procedures
ODP.NET does not support multiple stored procedures with the same name inside the configuration file. If an ODP.NET application uses an overloaded stored procedure, the application can store only one overloaded stored procedure information in the configuration file.

Type Initialization Exceptions
Type initialization exceptions can be caused by invalid .NET configuration file entries. Evaluate the exception that is caught as well as its inner exceptions to determine the .NET configuration file entry or the attribute setting that is causing the exception.

ODP.NET tracing logs the valid and invalid .NET configuration file entries that ODP.NET has parsed. To look for .NET configuration file related entries, set the TraceLevel to the Entry, exit, and SQL statement information level setting. Trace entries related to implicit REF CURSOR binding have a (REFCURSOR) entry along with (ERROR), if any errors are encountered.

Using Stored Functions with Function Import
Function Import only supports stored procedures, and does not support functions. When using the Add Function Import dialog for the Entity Data Model that you have created, the Get Column Information button does not return the metadata information for the REF CURSOR that is being returned by a stored function, even if it is configured properly in the .NET configuration file.

LOB Support
ODP.NET provides an easy and optimal way to access and manipulate large object (LOB) data types.

Note: SecureFiles can be used with existing ODP.NET LOB classes.
This section includes the following topics:

- Large Character and Large Binary Data Types
- 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 Data Types

Oracle Database supports large character and large binary data types.

**Large Character Data Types**

- **CLOB** - Character data can store up to 4 gigabytes.
- **NCLOB** - Unicode National character set data can store up to 4 gigabytes.

**Large Binary Data Types**

- **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 data types 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 manipulating LOB data: OracleBFile, OracleBlob, and OracleClob.

Table 3–17 shows the proper ODP.NET object to use for a particular Oracle LOB type.

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

Table 3–17  ODP.NET LOB Objects

The ODP.NET LOB objects can be obtained by calling the proper typed accessor on the OracleDataReader object, or by calling the proper typed accessor 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 BFILE types) 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 database round-trip. The OracleClob object 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.

Additional methods 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 of.

If an ODP.NET LOB object is obtained from an `OracleDataReader` object through a typed accessor, then its `Connection` property is set with a reference to the same `OracleConnection` object used by the `OracleDataReader` object. 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` object. 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 be used only within the context of the same `OracleConnection` referenced by the ODP.NET LOB object. For example, the ODP.NET LOB `Connection` property must reference the same connection as the `OracleCommand` object 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` object for automatically 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 .NET string data into a LOB column in an Oracle9i database or later, 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`.

Features of Oracle Data Provider for .NET 3-85
Updating LOBs Using ODP.NET LOB Objects

Oracle BFILES cannot be updated; therefore, 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 object before the command execution, so that the lock can be released when the OracleTransaction Commit or Rollback method is invoked.

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

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 the OracleDataReader typed accessors (GetOracleClobForUpdate or GetOracleBlobForUpdate) on the OracleDataReader object 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 object 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 BLOB, CLOB, and NCLOB objects. 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, an anonymous PL/SQL block, or a 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. Commit and rollback operations do not affect the data referenced by a temporary LOB.

ODP.NET XML Support

ODP.NET 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 is also allowed. Oracle Database supports XML natively in the database, through 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.

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
- Updating XML Data in OracleXmlType
- Characters with Special Meaning in XML
- Retrieving Query Result Set as XML
- Data Manipulation Using XML

## Supported XML Features

XML support in ODP.NET provides the ability to do the following:

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

**See Also:** "XQuery Support" on page 3-88

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

- Enhancements to the `OracleCommand`, `OracleConnection`, and `OracleDataReader` classes.
- The following XML-specific classes:
  - `OracleXmlType`
    
    `OracleXmlType` objects are used to retrieve Oracle native XMLType data.
  - `OracleXmlStream`
    
    `OracleXmlStream` objects are used to retrieve XML data from `OracleXmlType` objects as a read-only .NET Stream object.
  - `OracleXmlQueryProperties`
    
    `OracleXmlQueryProperties` objects represent the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Query`.
  - `OracleXmlSaveProperties`
    
    `OracleXmlSaveProperties` objects represent the XML properties used by the `OracleCommand` class when the `XmlCommandType` property is `Insert`, `Update`, or `Delete`.
XQuery Support

Beginning with Oracle Database 10g release 2(10.2), ODP.NET supports the XQuery language through a native implementation of SQL/XML functions, XMLQuery and XMLTable. When executing XQuery statements, Oracle XML DB generally evaluates XQuery expressions by compiling them into the same underlying structures as relational queries. Queries are optimized, leveraging both relational-database and XQuery-specific optimization technologies, so that Oracle XML DB serves as a native XQuery engine.

The treatment of all XQuery expressions, whether natively compiled or evaluated functionally, is transparent: programmers do not need to change their code to take advantage of XQuery optimizations.

See Also: Oracle XML DB Developer’s Guide to learn more about Oracle’s XQuery support

OracleXmlType and Connection Dependency

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

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

- Instantiated from an OracleDataReader class using the GetOracleXmlType, GetOracleValue, or GetOracleValues method:
  The Connection property is set with a reference to the same OracleConnection object used by the OracleDataReader object.

- 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

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 Oracle Data Provider for .NET in a few 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 the OracleDataAdapter.Update method is invoked, the OracleDataAdapter object ensures that the XMLType data is handled properly. The OracleDataAdapter object uses any custom SQL INSERT, UPDATE, or DELETE statements that are provided. Otherwise, valid SQL statements are generated by the OracleCommandBuilder object as needed to flush the changes to the database.

Updating with OracleCommand and OracleParameter

The OracleCommand class provides a powerful way of updating XMLType data, especially with the use of an OracleParameter object. 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 an OracleXmlType object, having an instance of an OracleXmlType class 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 object that a NULL value is to be inserted.
  Passing in a null OracleXmlType object as an input parameter does not insert a NULL value into the XMLType column. In this case, the OracleCommand object raises an exception.

- Setting NULL Values in an XMLType Column without input binding
  The following example demonstrates setting NULL values in an XMLType column without input binding:
// Create a table with an XMLType column in the database
CREATE TABLE XML_TABLE(NUM_COL number, XMLTYPE_COL xmltype);

An application can set a NULL value in the XMLType column by explicitly inserting
a NULL value or by not inserting anything into that column as in the following
examples:
insert into xml_table(xmltype_col) values(NULL);
update xml_table t set t.xmltype_col=NULL;

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

// Create a table with an XMLType column in the database
CREATE TABLE 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

The following are 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.

Characters with Special Meaning in XML

The following characters in Table 3–18 have special meaning in XML. For more
information, refer to the XML 1.0 specifications

<table>
<thead>
<tr>
<th>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. A blank space is not a valid character in an XML element name. If a SQL identifier contains a space character, then in the corresponding XML element name, the space character is replaced by \( _x0020_ \), which is based on Unicode encoding of the space character.

### Retrieving Query Result Set as XML

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

### Handling Date and Time Format

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

<table>
<thead>
<tr>
<th>Database Release</th>
<th>Date and Time Format Supported</th>
</tr>
</thead>
</table>
| Oracle9i release 2 (9.2.x) and Oracle Database 10g | Oracle DATE type data is retrieved in the format specified using the NLS\_DATE\_FORMAT in the session.  
TIMESTAMP and TIMESTAMP WITH TIME ZONE type data is retrieved in the format specified using the NLS\_TIMESTAMP\_FORMAT and the NLS\_TIMESTAMP\_T2\_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: YYYY-MM-DDThh:mm:ss.sss (ISO Format notation).  
To do this, before the query is executed, the application must explicitly perform an ALTER SESSION statement on the session for the following NLS session parameters:  
- NLS\_DATE\_FORMAT - Must be set to the following Oracle Date and Time Format: YYYY-MM-DD\*T\"HH24:MI:SS  
- NLS\_TIMESTAMP\_FORMAT - Must be set to the following Oracle Date and Time Format: YYYY-MM-DD\*T\"HH24:MI:SS.FF3  
- NLS\_TIMESTAMP\_T2\_FORMAT - Must be set to the following Oracle Date and Time Format: YYYY-MM-DD\*T\"HH24:MI:SS.FF3 |
| Oracle Database 10g release 2 (10.2) or later | The generated XML DATE and TIMESTAMP formats are based on the standard XML Schema formats.  
For more information on the XML Schema specification, see http://www.w3.org/TR/2004/REC-xmldatatypes-20041028/datatypes.html#isoformats |

### Characters with Special Meaning in Column Data

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

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

```csharp
/* Database Setup  
connect scott/tiger@oracle  
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: The XML entity encoding that represents the angle brackets appears in bold.

<?xml version = '1.0'?><ROWSET>
    <ROW>
        <id>1</id>
        <NAME>&lt;Jones&gt;</NAME>
    </ROW>
</ROWSET>
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 of 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, as shown in the following example:

```
// 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.

This section demonstrates how Oracle database handles the mapping of SQL identifiers to XML element names, when retrieving query results as XML from the
database. The demonstration uses the 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 column named some id that contains a blank space character. The space character is not allowed in an XML element name.

When retrieving the query results as XML, 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 is converted to a sequence of hexadecimal digits, derived from the Unicode encoding of the characters, bracketed by an introductory underscore, a lowercase x, and a trailing underscore.

Thus, 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:** "Characters with Special Meaning in XML" on page 3-90

### Improving Default Mapping

You can improve the default mapping of SQL identifiers to XML element names by using 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. Prefix the aliases with an at sign (@) to map them to XML attributes instead of XML elements.
- Modify the XML document. Use Extensible Stylesheet Language Transformation (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 may have an impact on 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, as shown in the following example:

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

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

**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 data using XML.

**Handling Date and Time Format**

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

<table>
<thead>
<tr>
<th>Database Release</th>
<th>Date and Time Format Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle9i release 2 (9.2.x) and Oracle Database 10g</td>
<td>All DATE, TIMESTAMP, and TIMESTAMP WITH TIME ZONE type data must be specified in the XML document in the ISO Date and Time Format YYYY-MM-DD'T'HH24:MI:SS.SSS (ISO Format notation). The following string is the ISO Date and Time Format notation represented in the Oracle Date and Time Format notation: YYYY-MM-DD'T'HH24:MI:SS.FF3. In addition to using the ISO Format notation in the XML document, before the save is executed, the application must explicitly perform an ALTER SESSION command on the session for the following NLS session parameters:</td>
</tr>
<tr>
<td></td>
<td>■ NLS_DATE_FORMAT - Must be set to the following Oracle Date and Time Format: YYYY-MM-DD'T'HH24:MI:SS</td>
</tr>
<tr>
<td></td>
<td>■ NLS_TIMESTAMP_FORMAT - Must be set to the following Oracle Date and Time Format: YYYY-MM-DD'T'HH24:MI:SS.FF3</td>
</tr>
<tr>
<td></td>
<td>■ NLS_TIMESTAMP_TZ_FORMAT - Must be set to the following Oracle Date and Time Format: YYYY-MM-DD'T'HH24:MI:SS.FF3</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 are insert, update, or delete changes.

The insert change must be in an XML document containing only rows to be inserted, the update changes only with rows to be updated, and the delete changes 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>
```

Characters with Special Meaning in Column Data

If the data in any of the elements in the XML document contains characters that have a special meaning in XML (see Table 3–18), these characters must be replaced with appropriate entity encoding, or be preceded by an escape character in the XML document, so that the data is stored correctly in the database table column. Otherwise, ODP.NET throws an exception.

The following example demonstrates how ODP.NET handles the angle bracket special characters in the column data, using entity encoding:

```sql
// Create the following table
create table specialchars (*id* number, name varchar2(255));
```
The following XML document can be used to insert values (1, ‘<Jones>’) into the specialchars table. The XML entity encoding that represents the angle brackets appears in bold.

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

**Characters with Special Meaning 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 that represents a row of data in the XML document, the child XML elements map to database column names. The mapping of the child element name to the column name is always case-sensitive, but the root tag and row tag names are case-insensitive. The following example demonstrates this case-sensitivity:

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

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

```xml
<?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 of the column name.

**XML Element Name to Column Name Mapping**

This section describes how Oracle database handles the mapping of XML element names to column names when using XML for data manipulation in the database. The following specialchars table involving the some id column demonstrates this handling.

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

Note that the specialchars table has a column named some id that contains a blank space character. The space character is not allowed in an XML element name.
Saving Changes to a Table Using an XML Document

When an XML document is used to save changes to a table or view, the `OracleCommand.XmlSaveProperties.UpdateColumnsList` property is used to specify the list of columns to update or insert.

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 that are not valid in an XML element name, the escaped column name must be specified in the `UpdateColumnsList` property as in the following example.

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

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

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

```csharp
/* Database Setup
connect scott/tiger@oracle
drop table specialchars;
create table specialchars ('some 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 InsertUsingXmlDocSample
{
static void Main()
{
    OracleConnection con = new OracleConnection();
    con.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;\n";
    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;

    // Set the XML document
    cmd.CommandText = "<?xml version = '1.0'?>\n" + "<ROWSET>\n" + "<ROW>\n" + "<some_x0020_id>2</some_x0020_id>\n" + "<NAME>&lt;Jones&gt;</NAME>\n" + "</ROW>\n" + "</ROWSET>\n";
    cmd.XmlSaveProperties.Table = 'specialchars';

    string[] ucols = new string[2];
ucols[0] = 'some_x0020_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** You can improve the default mapping by using 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 may have an impact on 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 save changes to multiple relational tables that have been joined together. Oracle recommends that you create a view on those relational tables, and then update that view. If the view cannot be updated, triggers can be used instead.

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

**Commit Transactions**
When the changes in an XML document are made, either all the changes are committed, or if an error occurs, all changes are rolled back.

**Oracle User-Defined Types (UDTs) and .NET Custom Types**
ODP.NET has the ability to represent Oracle UDTs found in the database as custom types in .NET applications. UDTs are useful in representing complex entities as a single object that can be shared among applications. Oracle products, such as Oracle Spatial and Oracle XML DB, use their own complex types frequently.
To represent Oracle UDTs as .NET custom types, applications must apply .NET attributes to custom classes and structs, and to their public fields and properties. To convert between UDTs and custom types, ODP.NET uses custom interfaces. This section discusses the following topics:

- Oracle User-Defined Types (UDTs)
- Custom Types
- Specifying Custom Type Mappings
- Converting Between Custom Types and Oracle UDTs
- Oracle UDT Attribute Mappings
- Oracle UDT Retrieval from OracleDataReader
- Oracle UDT Metadata Retrieval from OracleDataReader
- Oracle UDT Parameter Binding with OracleParameter
- Populating the DataSet with Oracle UDTs
- UDT Method Invocation
- Configuration Settings for Oracle UDTs

**Oracle User-Defined Types (UDTs)**

Oracle Data Provider for .NET supports Oracle object types or user-defined types (UDTs), which are defined in the Oracle database.

There are two kinds of UDTs:

- Object types (Oracle Object)
- Collection types (which can be VARRAY types or nested table types)

Additionally, ODP.NET supports references (REF) to object types.

**See Also:** "OracleRef Class" on page 16-51

The term UDT is used interchangeably with Oracle object types and abstract data types (ADTs).

**See Also:** Oracle Database Application Developer’s Guide - Object-Relational Features for complete descriptions of object types

The name of the Oracle UDT is case-sensitive and must be in the form `schema_name.type_name`.

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

**Custom Types**

Oracle Data Provider for .NET supports UDTs by representing Oracle UDTs defined in the database as .NET types, that is, custom types. For every Oracle UDT that the application wishes to fetch and manipulate, one custom type factory and one custom type are needed. The custom factory class is solely responsible for instantiating the custom type. ODP.NET uses the interfaces implemented on the custom factory classes to instantiate custom types at run time. Custom types define the mapping between the...
Oracle UDT attributes or elements to the .NET members. ODP.NET uses the interfaces implemented on the custom type instances to transfer values between the Oracle UDT and the custom type at run time.

Custom types can be .NET classes or structures. They can represent either Oracle Objects or Oracle Collections. Custom types can be implemented manually by the application developer or generated through an ODP.NET code generation tool.

Once the factory class and the custom type are defined and meet the implementation requirements, the application may set ODP.NET to automatically discover the mapping between the Oracle UDT and the custom type. This discovery process is based on the attribute that is applied on the custom factory class. Alternatively, the application can provide an explicit mapping through a configuration file.

Oracle Collections can be represented as an array of .NET Types. For example, an Oracle Collection type of `NUMBER` can be mapped to an `int[]`. Moreover, an Oracle Collection type of an Oracle UDT can be mapped to an array of the custom type.

Custom types must adhere to certain requirements in order for ODP.NET to represent Oracle UDTs as custom types. These requirements are as follows:

**Required Custom Type Implementations**

This section lists the required implementations for a custom .NET class or structure.

- **Oracle.DataAcess.Types.IOracleCustomType** interface implementation
  
  This interface is used for conversions between custom types and Oracle UDTs.

  The interface methods are implemented using the static methods of the `OracleUdt` class.

- **Custom Type Factories**

  A custom type factory is used to create an instance of a custom type. A custom type factory is an implementation of either the `IOracleCustomTypeFactory` interface, the `IOracleArrayTypeFactory` interface, or both interfaces, as follows:

  - To create a custom type that represents an Oracle Object, the custom type or a separate custom type factory class must implement the `Oracle.DataAcess.Types.IOracleCustomTypeFactory` interface.

  - To create a custom type that represents an Oracle Collection, the custom type or a separate custom type factory class must implement the `Oracle.DataAcess.Types.IOracleCustomTypeFactory` interface and the `Oracle.DataAcess.Types.IOracleArrayTypeFactory` interface.

  - To create an array type that represents an Oracle Collection, a custom type factory class must implement the `Oracle.DataAcess.Types.IOracleArrayTypeFactory` interface.

- **Custom Type Member Mapping Attributes**

  The custom type member mapping attributes specify the mapping between custom type members and either Oracle object attributes or Oracle collection elements.

  There are two types of custom type member mapping attributes:

  - `OracleObjectMappingAttribute`
This attribute specifies the mapping between custom type members and Oracle object attributes for custom types that represent Oracle objects. This attribute must be applied to each custom type member (either field or property) that represents an Oracle Object attribute.

Note: Not all Oracle object attributes need to be mapped to custom type members. If there is no OracleObjectMappingAttribute for a particular object attribute, ODP.NET ignores that object attribute when converting between Oracle objects and custom types.

- OracleArrayMappingAttribute
  This attribute specifies the custom type member that stores the elements of an Oracle collection for custom types representing Oracle collections.
  The attribute must be specified on only one of the custom type members.

- Oracle.DataAcess.Types.INullable interface implementation
  This interface is used to determine if an instance of a custom type represents a null UDT.
  The IsNull property of the interface enables applications and ODP.NET to determine whether or not the UDT is null.

- Static Null field
  The public static Null property is used to return a null UDT.
  This property returns a custom type with an IsNull property that returns true.

Optional Custom Type Implementations
The following are optional:

- IXMLSerializable
  The IXMLSerializable interface is used in the .NET 2.0 framework to enable conversion between the custom type and its XML representation.
  This interface is only used if the serialization and deserialization of a custom type is needed in the DataSet.

- Static Parse and Public ToString methods
  These methods enable conversion between the custom type and its string representation.
  These methods are invoked when a DataGrid control is used to accept changes and display instance values.

- Type Inheritance
  Type Inheritance refers to the process of deriving an Oracle UDT in the database from a super type.
  If the custom type represents an Oracle UDT that is derived from a super type, the custom class should follow the same type hierarchy, that is, the custom class should be derived from another custom class that represents the super type defined in the database.

- OracleCustomTypeMappingAttribute

Note: Not all Oracle object attributes need to be mapped to custom type members. If there is no OracleObjectMappingAttribute for a particular object attribute, ODP.NET ignores that object attribute when converting between Oracle objects and custom types.
The **OracleCustomTypeMappingAttribute** object specifies the mapping between a custom type (or an array type) and an Oracle UDT.

There must be a unique custom type factory for each Oracle UDT used by the application as follows:

- **Oracle Object Types:**
  
  The custom type factory must return a custom type that only represents the specified Oracle Object Type.

- **Oracle Collection Types:**
  
  The custom type factory may return a custom type that can be used by other Oracle Collection Types. This is common when an array type is used to represent an Oracle Collection, for example, when an `int[]` is used to represent a collection of `NUMBER`s.

If the **OracleCustomTypeMappingAttribute** is not specified, then custom type mappings must be specified through XML configuration files, that is, `machine.config`, and either `app.config` for Windows applications or `web.config` for web applications.

**See Also:**

- "**IOracleCustomType Interface**" on page 16-26
- "**OracleRef Class**" on page 16-51
- "**IOracleCustomTypeFactory Interface**" on page 16-30

### Specifying Custom Type Mappings

After creating a custom type, the application must specify a custom type mapping that maps the custom type to an Oracle UDT in the database. This can be done using a custom type factory or XML in configuration files.

Using XML to specify custom type mappings has priority, if both techniques have been implemented. At run time, if ODP.NET finds custom type mappings specified in configuration files, it ignores any custom type mappings specified through the **OracleCustomTypeMappingAttribute** object.

Custom type mappings cannot be specified using synonyms, regardless of whether or not the mapping is provided through the **OracleCustomTypeMappingAttribute** object or the XML configuration file.

**See Also:** Oracle Developer Tools for Visual Studio help sections on User-Defined Types Node, under Server Explorer for Visual Studio 2005 and Oracle Explorer for Visual Studio 2003, for further information on UDT mapping

This section contains these topics:

- "Using a Custom Type Factory to Specify Custom Type Mappings"
- "Using XML in Configuration Files to Specify Custom Type Mappings"

### Using a Custom Type Factory to Specify Custom Type Mappings

The application can specify a custom type mapping using a custom type factory. The application supplies the name of the Oracle UDT, in the format `schema_name.type_name`, to an **OracleCustomTypeMappingAttribute** object and applies the name to the corresponding custom type factory. A custom type factory is a class or struct that
implements either or both the IOracleCustomTypeFactory and IOracleArrayTypeFactory interfaces.

Note that for each Oracle UDT used by the application, there must be a unique custom type factory. Additionally, for Oracle Object Types, the custom type factory must return a custom type that uniquely represents the specified Oracle Object Type. For Oracle Collection Types, the custom type factory returns a custom type that can be used by other Oracle Collection Types. This is common when an custom type that is an array type represents an Oracle Collection, that is, when an int[] is used to represent a collection of NUMBERs.

At run time, using reflection programming, ODP.NET discovers all the custom type mappings specified by the application through the OracleCustomTypeMappingAttribute object.

---

**Note:** The UDT name that is specified in the OracleCustomTypeMappingAttribute may not contain a period.

---

**Using XML in Configuration Files to Specify Custom Type Mappings**

The application can specify a custom type mapping with XML in configuration files, for example: using machine.config, and either app.config for Windows applications or web.config for web applications.

The custom type mappings must be specified in the oracle.dataaccess.client configuration section group. Each custom type mapping must be added to the collection of custom type mappings using the XML element `<add>`.

Each custom type mapping is consists of a name attribute and a value attribute. The name attribute may be any user-specified name that represents the custom type mapping. The value attribute must begin with `udtMapping` and be followed by the required and optional attributes listed below.

**Required Attributes**

- **factoryName**
  
  The case-sensitive assembly qualified name of the custom type factory class or struct.
  
  If the assembly that defines the custom type factory does not have a strong name, then a partial assembly name consisting of just the assembly name is sufficient. In the case of strongly named assemblies, a complete assembly name is required. It must include the assembly name, the `Version`, `Culture`, `PublicKeyToken`.

- **typeName**
  
  The case-sensitive name of the UDT defined in the database. By default all UDTs are created in the database with upper case names

- **schemaName**
  
  The case-sensitive schema in which the UDT is defined in the database. By default all schemas are created in the database with upper case names

**Optional Attributes**

- **dataSource**
If specified, indicates that the custom type mapping applies only to Oracle UDTs defined in the database that the application connects to, as specified by the TNS name alias.

The Data Source is case-insensitive.

The following is an example of the format of the XML that can be specified in the configuration file for .NET 2.0:

```xml
<oracle.dataaccess.client>
  <settings>
    <add name="Person" value="udtMapping factoryName='Sample.PersonFactory,
Sample, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null'
typeName='PERSON' schemaName='SCOTT' dataSource='oracle'/">
    <add name="Student" value="udtMapping factoryName='Sample.StudentFactory,
Sample, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null'
typeName='STUDENT' schemaName='SCOTT'/">
  </settings>
</oracle.dataaccess.client>
```

### Using Custom Type Mappings

During data retrieval, the application uses the custom type mappings to convert an Oracle UDT to a custom type. When data is provided back to the database through an input or input/output parameter, or by an update through an Oracle REF, the application uses the mappings to convert the custom type to an Oracle UDT.

In the case of input and input/output parameters, the application must also set the `OracleParameter UdtTypeName` property to the user-defined type name of the parameter.

In certain cases, where Oracle UDTs are part of a type hierarchy, the custom type must be instantiated as a specific type in the type hierarchy. The Oracle UDT provided by the custom type mapping must a subtype of the Oracle UDT specified by the `OracleParameter UdtTypeName` property.

For example, the parameter for a stored procedure is of type, `SCOTT.PERSON` and has a subtype, `SCOTT.STUDENT`. The application has a custom class instance that represents `SCOTT.STUDENT`. The `UdtTypeName` is set to `SCOTT.PERSON`, but the custom type mapping indicates that the custom class is mapped to `SCOTT.STUDENT` and overrides the `UdtTypeName` when it instantiates the Oracle UDT. Thus, ODP.NET instantiates and binds Oracle UDTs appropriately when the custom object represents an Oracle UDT that is a subtype of the parameter type.

### Converting Between Custom Types and Oracle UDTs

ODP.NET can convert between Oracle UDTs and custom types, if the proper attribute mappings are specified and the custom types are defined properly.

ODP.NET performs a conversion whenever an Oracle UDT is fetched as:

- In, out, in/out parameters bound for SQL or PL/SQL execution

  The `DbType` property of `OracleParameter` must be set to `DbType.Object` or the `OracleDbType` property must be set to `OracleDbType.Object` or `OracleDbType.Array`.

  For parameters that are user-defined types, the `UdttypeName` property of the `OracleParameter` object must be always set to the parameter type.
Note: The UdtTypeName may differ from the Oracle UDT specified in the custom type mapping. This is the case when the parameter type is a super type of the Oracle UDT that the custom type represents.

- Column value retrieved from an OracleDataReader object

If the application requests for the value either through the GetValue, GetValues, GetOracleValue, GetOracleValues, GetProviderSpecificValue, or GetProviderSpecificValues methods or the Item[] property for a UDT column, ODP.NET finds the corresponding custom type that represents the Oracle UDT and carries out the proper conversion.

- Part of a Resultset that populates the DataSet

If the application populates the DataSet with a result that contains UDTs using the Fill method on the OracleDataAdapter, the DataSet is populated with custom types that represent Oracle UDTs. With ADO.NET 2.0, the DataSet is populated with custom types for UDT columns regardless of whether the ReturnProviderSpecificTypes on the OracleDataAdapter is set to true or false.

- A Object referenced through a REF

When an Object referenced by a REF is retrieved, the custom type that represents the Oracle UDT is returned.

The application can use the OracleUdtFetchOption method to control the copy of the Object that is returned as follows:

- If the OracleUdtFetchOption.Cache option is specified and a cached copy of the object exists, the cached copy is immediately returned. If no cached copy exists, the latest object copy from the database is cached and returned.

- If the OracleUdtFetchOption.Server option is specified, the latest object copy from the database is cached and returned. If the object is already cached, the latest object copy overwrites the existing one.

- If the OracleUdtFetchOption.TransactionCache option is specified, there are two possibilities within the same transaction:
  - If the object copy was previously retrieved using the Server or TransactionCache option, the TransactionCache option behavior becomes equivalent to the Cache option behavior.
  - If the object copy was not previously retrieved using the Server or TransactionCache option, the TransactionCache option behavior becomes equivalent to the Server option behavior.

Oracle UDT Attribute Mappings

Table 3–21 lists valid mappings of attributes (for objects) and elements (for collections), between Oracle UDT types and custom object types which can be either .NET types or Oracle provider-specific types (ODP.NET types).

Oracle collections do not have to map to a custom class. They can map to arrays of a specific type. Table 3–21 indicates those collections with elements of a specified Oracle type that can map to arrays of a .NET Type or a provider-specific type. For example, if an Oracle Collection is a VARRAY of NUMBER(8), it can map to a typeof(int[]). This eliminates the need to construct a class that only holds an int[].

Oracle User-Defined Types (UDTs) and .NET Custom Types

3-106 Oracle Data Provider for .NET Developer's Guide
For .NET 2.0, Oracle Collections can be mapped to Nullable types. This allows .NET 2.0 applications to obtain a nullable `int[]` which can hold null values in the `int[]`.
Note that Oracle UDT attributes and elements cannot be mapped to `object` or `object[]`.

<table>
<thead>
<tr>
<th>Type of UDT Attribute or Element</th>
<th>.NET Type</th>
<th>ODP.NET Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFILE #1</td>
<td>System.Byte[]</td>
<td>OracleBFile</td>
</tr>
<tr>
<td>BLOB</td>
<td>System.Byte[]</td>
<td>OracleBlob</td>
</tr>
<tr>
<td>CHAR</td>
<td>System.Char[], System.String</td>
<td>OracleString</td>
</tr>
<tr>
<td>CLOB</td>
<td>System.Char[], System.String</td>
<td>OracleClob</td>
</tr>
<tr>
<td>DATE</td>
<td>System.DateTime</td>
<td>OracleDate</td>
</tr>
<tr>
<td>INTERVAL DAY TO SECOND</td>
<td>System.TimeSpan,</td>
<td>OracleIntervalDS</td>
</tr>
<tr>
<td>INTERVAL YEAR TO MONTH</td>
<td>System.Int64</td>
<td>OracleIntervalYM</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>System.Byte[]</td>
<td>OracleBinary</td>
</tr>
<tr>
<td>NCHAR</td>
<td>System.Char[], System.String</td>
<td>OracleString</td>
</tr>
<tr>
<td>NCLOB</td>
<td>System.Char[], System.String</td>
<td>OracleClob</td>
</tr>
<tr>
<td>Nested Table</td>
<td><code>custom type, .NET type[], or custom type[]</code></td>
<td><code>ODP Type[]</code></td>
</tr>
<tr>
<td>NVARCHAR2</td>
<td>System.Char[], System.String</td>
<td>OracleString</td>
</tr>
<tr>
<td>Object Type</td>
<td><code>custom type</code></td>
<td>N/A</td>
</tr>
<tr>
<td>RAW</td>
<td>System.Byte[]</td>
<td>OracleBinary</td>
</tr>
<tr>
<td>REF</td>
<td>System.String</td>
<td>OracleRef</td>
</tr>
</tbody>
</table>
Oracle User-Defined Types (UDTs) and .NET Custom Types

Table 3–21 (Cont.) Attribute Mappings Between UDTs and Custom Object Types

<table>
<thead>
<tr>
<th>Type of UDT Attribute or Element</th>
<th>.NET Type</th>
<th>ODP.NET Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>System.DateTime</td>
<td>OracleTimeStamp</td>
</tr>
<tr>
<td>TIMESTAMP WITH LOCAL TIME ZONE</td>
<td>System.DateTime</td>
<td>OracleTimeStampL TZ</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>System.DateTime</td>
<td>OracleTimeStampTZ</td>
</tr>
<tr>
<td>VARCHAR2</td>
<td>System.Char[], System.String</td>
<td>OracleString</td>
</tr>
<tr>
<td>VARRAY</td>
<td>custom type, .NET type[], or custom type[]</td>
<td>ODP Type[]</td>
</tr>
</tbody>
</table>

Notes:
1. Conversion from a System.Byte[] to a BFILE is not supported, and therefore, System.Byte[] only represents a BFILE in read-only scenarios.

Oracle UDT Retrieval from OracleDataReader

In order to retrieve Oracle UDTs from the OracleDataReader, an application must specify a custom type mapping that determines the type that will represent the Oracle UDT. Once a custom type mapping has been specified and any necessary custom types have been created, the application can retrieve Oracle UDTs.

Table 3–22 shows the type and value returned from an OracleDataReader object based on the method invoked, the column type, and whether or not there is a valid Custom type mapping.

Note: PS Object refers to a provider-specific object.

Table 3–22 Type and Value Returned from OracleDataReader Object

<table>
<thead>
<tr>
<th>OracleDataReader method/property invocation</th>
<th>Column Data Type</th>
<th>Custom Type Mapping</th>
<th>Value Returned for Oracle UDT</th>
<th>NULL Value Returned for Oracle UDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item[index], Item[name], GetValue(), GetValues()</td>
<td>Object, Collection</td>
<td>none</td>
<td>Exception thrown</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>Item[index], Item[name], GetValue(), GetValues()</td>
<td>Object</td>
<td>schema.type</td>
<td>custom object</td>
<td>DBNull.Value</td>
</tr>
<tr>
<td>Item[index], Item[name], GetValue(), GetValues()</td>
<td>Collection</td>
<td>schema.type</td>
<td>custom object</td>
<td>.NET Type[]</td>
</tr>
<tr>
<td>Item[index], Item[name], GetValue(), GetValues()</td>
<td>REF</td>
<td>none</td>
<td>string(HEX)</td>
<td>DBNull.Value</td>
</tr>
</tbody>
</table>
Oracle UDT Metadata Retrieval from OracleDataReader

An OracleDataReader object can return metadata used to determine the custom type that represents an Oracle UDT when a .NET Type or Provider-Specific Type accessor is invoked. The same custom type is used when populating the DataSet using the OracleDataAdapter.Fill method.

Table 3–23 shows the values returned from the OracleDataReader GetFieldType and GetProviderSpecificFieldType methods that specify the .NET type of the column.

### Table 3–23 Values Returned from OracleDataReader Methods

<table>
<thead>
<tr>
<th>OracleDataReader Method/Property Invocation</th>
<th>Column Data Type</th>
<th>Custom Type Mapping</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFieldType(index)</td>
<td>Object, Collection</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>GetFieldType(index)</td>
<td>Object</td>
<td>schema.type</td>
<td>typeof(custom type)</td>
</tr>
<tr>
<td>GetFieldType(index)</td>
<td>Collection</td>
<td>schema.type</td>
<td>typeof(custom type)</td>
</tr>
<tr>
<td>GetFieldType(index)</td>
<td>REF</td>
<td>none</td>
<td>typeof(string)</td>
</tr>
</tbody>
</table>

See Also: "Obtaining Data from an OracleDataReader Object" on page 3-62
Oracle User-Defined Types (UDTs) and .NET Custom Types

Oracle User-Defined Types (UDTs) and .NET Custom Types

This section discusses using UDT output and input parameter bindings with an OracleParameter object.

See Also: "Parameter Binding" on page 3-45

This section contains these topics:

- Guidelines for Binding UDT Input and Output Parameters
- UDT Input Parameter Binding with OracleParameters
- UDT Output Parameter Binding with OracleParameters

Guidelines for Binding UDT Input and Output Parameters

Developers must consider the following when using UDT parameter bindings with an OracleParameter object.

- The UdtTypeName property must be set. Binding is based on the UdtTypeName property regardless of the parameter direction.

Note: The UdtTypeName may differ from the Oracle UDT specified in the custom type mapping. This occurs when the parameter type is a super type of the Oracle UDT that the custom type represents.

- In case of Input/Output binding, the behavior is the same as Input and Output parameters.
- For Input parameter values, the bind value is converted to the UDT specified by the custom type mapping.
- For Output parameters:
  - If the value being returned is an Oracle Object or Collection, it is converted to a custom type or array type as specified by the custom type mapping. The value returned is always a custom type or an array type, regardless of whether the property most recently set was DbType or OracleDbType.
  - If the value being returned is a REF, then no custom type mapping is required.

Table 3–23 (Cont.) Values Returned from OracleDataReader Methods

<table>
<thead>
<tr>
<th>OracleDataReader Method/Property Invocation</th>
<th>Column Data Type</th>
<th>Custom Type Mapping</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetProviderSpecificField(index)</td>
<td>Object</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td></td>
<td>Collection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetProviderSpecificField(index)</td>
<td>Object</td>
<td>schema.type</td>
<td>typeof(custom type)</td>
</tr>
<tr>
<td></td>
<td>Collection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetProviderSpecificField(index)</td>
<td>Collection</td>
<td>schema.type</td>
<td>typeof(custom type)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>typeof(OracleRef)</td>
</tr>
</tbody>
</table>

Oracle UDT Parameter Binding with OracleParameter

This section discusses using UDT output and input parameter bindings with an OracleParameter object.

See Also: "Parameter Binding" on page 3-45

This section contains these topics:

- Guidelines for Binding UDT Input and Output Parameters
- UDT Input Parameter Binding with OracleParameters
- UDT Output Parameter Binding with OracleParameters

Guidelines for Binding UDT Input and Output Parameters

Developers must consider the following when using UDT parameter bindings with an OracleParameter object.

- The UdtTypeName property must be set. Binding is based on the UdtTypeName property regardless of the parameter direction.

Note: The UdtTypeName may differ from the Oracle UDT specified in the custom type mapping. This occurs when the parameter type is a super type of the Oracle UDT that the custom type represents.

- In case of Input/Output binding, the behavior is the same as Input and Output parameters.
- For Input parameter values, the bind value is converted to the UDT specified by the custom type mapping.
- For Output parameters:
  - If the value being returned is an Oracle Object or Collection, it is converted to a custom type or array type as specified by the custom type mapping. The value returned is always a custom type or an array type, regardless of whether the property most recently set was DbType or OracleDbType.
  - If the value being returned is a REF, then no custom type mapping is required.
UDT Input Parameter Binding with OracleParameters

Only certain combinations of these OracleParameter property values, DbType, OracleDbType, and UdtTypeName, can exist on the OracleParameter object. OracleParameter objects cannot be set to combinations that are not listed.

Table 3–24 describes the valid ways of binding input parameters for Oracle UDTs.

The last column indicates the Oracle type that ODP.NET converts the OracleParameter value to before binding.

Table 3–24 Valid Ways to Bind Input Parameters for Oracle UDTs

<table>
<thead>
<tr>
<th>OracleParameter. Value</th>
<th>OracleParameter. DbType or OracleDbType</th>
<th>OracleParameter . UdtTypeName</th>
<th>Custom Type Mappings</th>
<th>Oracle Type converted to before Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom object</td>
<td>DbType.Object</td>
<td>not set</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>custom object[]</td>
<td>DbType.Object</td>
<td>not set</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>.NET object[]</td>
<td>DbType.Object</td>
<td>not set</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>custom object[]</td>
<td>DbType.Object</td>
<td>not set</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>custom object</td>
<td>DbType.Object</td>
<td>schema.type</td>
<td>schema.type</td>
<td>Specified UDT is instantiated. Value is bound as Object or Collection, based on the UdtTypeName property</td>
</tr>
<tr>
<td>custom object</td>
<td>OracleDbType.Object</td>
<td>schema.type</td>
<td>schema.type</td>
<td>Specified UDT is instantiated. schema.type must represent an object.</td>
</tr>
<tr>
<td>custom object</td>
<td>OracleDbType.Array</td>
<td>schema.type</td>
<td>schema.type</td>
<td>Specified UDT is instantiated. schema.type must represent a collection.</td>
</tr>
<tr>
<td>.NET object[]</td>
<td>DbType.Object</td>
<td>schema.type</td>
<td>schema.type</td>
<td>UDT specified by OracleParameter.UdtTypeName is instantiated.</td>
</tr>
<tr>
<td>.NET object[]</td>
<td>OracleDbType.Object</td>
<td>schema.type</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>.NET object[]</td>
<td>OracleDbType.Object</td>
<td>schema.type</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
</tbody>
</table>
Only certain combinations of these `OracleParameter` property values, `DbType`, `OracleDbType`, and `UdtTypeName`, can exist on the `OracleParameter` object. `OracleParameter` objects cannot be set to combinations that are not listed.

Table 3–25 shows the supported ODP.NET output parameter bindings of Oracle database objects.

The last column indicates the type that ODP.NET converts the `OracleParameter` value to before binding.

### UDT Output Parameter Binding with `OracleParameters`

Table 3–24 (Cont.) Valid Ways to Bind Input Parameters for Oracle UDTs

<table>
<thead>
<tr>
<th>OracleParameter. Value</th>
<th><code>OracleDbType</code></th>
<th><code>UdtTypeName</code></th>
<th>Custom Type Mappings</th>
<th>Oracle Type converted to before Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom object</td>
<td><code>OracleDbTypeRef</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td><code>[].NET object[]</code></td>
<td><code>OracleDbTypeObject</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td><code>PS object[]</code></td>
<td><code>OracleDbTypeObject</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>custom object</td>
<td><code>OracleDbTypeObject</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td><code>String(HEX)</code></td>
<td><code>OracleDbTypeObject</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td><code>OracleString(HEX)</code></td>
<td><code>OracleDbTypeObject</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td><code>OracleRef</code></td>
<td><code>OracleDbTypeArray</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>A REF</td>
</tr>
</tbody>
</table>

### Table 3–25 Valid Ways to Bind Output Parameters for Oracle UDTs

<table>
<thead>
<tr>
<th>Type returned from Oracle</th>
<th><code>OracleParameter. DbType</code></th>
<th><code>UdtTypeName</code></th>
<th>Custom Type Mappings</th>
<th>Type converted to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object/Collection/REF</td>
<td><code>DbTypeObject</code></td>
<td><code>not set</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>Object/Collection</td>
<td><code>DbTypeObject</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>Object</td>
<td><code>DbTypeObject</code></td>
<td><code>schema.type</code></td>
<td><code>schema.type</code></td>
<td>custom object</td>
</tr>
<tr>
<td>Object</td>
<td><code>DbTypeArray</code></td>
<td><code>schema.type</code></td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>Collection</td>
<td><code>DbTypeArray</code></td>
<td><code>schema.type</code></td>
<td><code>schema.type</code></td>
<td>custom object</td>
</tr>
</tbody>
</table>
Populating the DataSet with Oracle UDTs

The DataSet is a disconnected result set. With ADO.NET 2.0, both .NET types and provider-specific types can be used to populate the DataSet. This section describes the types used to populate the DataSet when the column is an Oracle UDT.

Table 3–26 lists the types that populate the DataSet column, based on the Oracle column type, the ReturnProviderSpecificTypes property of the DataAdapter, the existence of a custom type mapping, the DataSet column type, the DataSet column value, and the DataSet column null value.

<table>
<thead>
<tr>
<th>Type returned from Oracle</th>
<th>OracleDbType</th>
<th>OracleParameter. DbType</th>
<th>UdtTypeName</th>
<th>Custom Type Mappings</th>
<th>Type converted to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection</td>
<td>OracleDbType</td>
<td>OracleDbType.Object</td>
<td>schema.type</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>REF</td>
<td>DbType.Object</td>
<td>OracleDbType.Object</td>
<td>schema.type</td>
<td>none</td>
<td>Exception thrown</td>
</tr>
<tr>
<td>REF</td>
<td>OracleDbType</td>
<td>OracleDbType.Array</td>
<td>schema.type</td>
<td>none</td>
<td>OracleRef</td>
</tr>
</tbody>
</table>

See Also:
- "Parameter Binding" on page 3-45
- "Typed OracleDataReader Accessors" on page 3-62
**UDT Method Invocation**

ODP.NET supports invocation of methods defined for a UDT on the database. This can be accomplished by doing the following:

1. Set the `CommandType` as `CommandType.StoredProcedure`.
2. Set the `CommandText` as "`type_name.procedure_name`".
3. Execute the command using any of the `Execute` methods on the `OracleCommand` object.

For instance functions, the parameters are as follows:

- The first parameter must be the return value.
- The second parameter must be the UDT instance on which the instance method is invoked, which is the instance of the .NET custom object.
- Subsequent parameters are for the function.

For instance procedures, the first parameter must be the UDT instance.

For static methods, the UDT instance is not needed.

**Configuration Settings for Oracle UDTs**

ODP.NET exposes two configuration settings to determine how ODP.NET handles Oracle UDTs.

- `StatementCacheWithUdts`
- `UdtCacheSize`

These configuration settings can be specified as machine-wide settings for a particular version of ODP.NET, using the registry key with the name that exists under `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ODP.NET\Assembly_Version`. The configuration settings specified in the registry can be overridden if an entry is created in the `machine.config` for .NET framework-wide settings, or in the `app.config` or `web.config` for application-specific settings. For details on configuring ODP.NET, see "Configuring Oracle Data Provider for .NET" on page 2-5.

**StatementCacheWithUdts**

`StatementCacheWithUdts` specifies whether or not ODP.NET caches Oracle UDTs retrieved by a `SELECT` statement along with the statement when it is returned to the statement cache. Possible values are 1 - Yes (the default) or 0 - No.

For the value of 1, the Oracle UDTs are cached along with the statements. Therefore, the memory that contained the UDTs can be re-used; subsequent executions of the same statement do not require additional memory. This may result in an overall higher performance.

For the value of 0, ODP.NET frees the memory for the retrieved Oracle UDTs before the statement is returned to the statement cache. This may result in poorer performance because subsequent executions will require new memory allocations.

**UdtCacheSize**

`UdtCacheSize` specifies the size of the object cache for each connection that ODP.NET uses when retrieving and manipulating Oracle UDTs. The value for this setting must be specified in kilobytes (KB) with the default 4096KB, equivalent to 4 MB.
This configuration setting is used to determine how frequently the objects in the object cache will be purged (using an LRU approach) as the limit of the object cache size approaches.

Oracle Streams Advanced Queuing Support

Oracle Streams Advanced Queuing (AQ) provides database-integrated message queuing functionality. Oracle Streams AQ is built on top of Oracle Streams and leverages the functions of Oracle Database so that messages can be stored persistently, propagated between queues on different computers and databases, and transmitted using Oracle Net Services and HTTP(S).

See Also: Oracle Streams Advanced Queuing User’s Guide and Reference

As Oracle Streams AQ is implemented in database tables, all operational benefits of high availability, scalability, and reliability are also applicable to queue data. Oracle Streams AQ supports standard database features such as recovery, restart, and security.

The following items discuss Oracle Streams AQ concepts:

- Queues and Queue Tables
  
  Messages enqueued in a queue are stored in a queue table. A queue table must be created before creating a queue based on it. Use the DBMS_AQADM PL/SQL package or Oracle Developer Tools for Visual Studio to create and administer queue tables and queues.

  Queues are represented by OracleAQQueue objects.

  See Also: OracleAQQueue Class on page 12-48

- Single-Consumer and Multiple-Consumer Queues
  
  A single-consumer queue is created based on a single consumer queue table. Messages enqueued in a single-consumer queue can be dequeued by only a single consumer.

  A multiple-consumer queue is based on a multiple-consumer queue table. This queue supports queue subscribers and message recipients.

- Message Recipients
  
  A message producer can submit a list of recipients when enqueuing a message. This allows for a unique set of recipients for each message in the queue. The recipient list associated with the message overrides the subscriber list, if any, associated with the queue. The recipients need not be in the subscriber list. However, recipients can be selected from among the subscribers.

  The Recipients property of an OracleAQMessage can be used to specify the recipients to a specific message in terms of OracleAQAgent objects.

  See Also: Recipients on page 12-35, OracleAQAgent Class on page 12-2

- Enqueue
Messages are enqueued when producer applications push the messages into a queue. This is accomplished by calling the `Enqueue` method on an `OracleAQQueue` object. Multiple messages can be enqueued using the `EnqueueArray` method.

See Also:
- "Enqueue" on page 12-68
- "EnqueueArray" on page 12-70

Dequeue

Messages are dequeued when consumer applications pull the messages from a queue. This is accomplished by calling the `Dequeue` method on an `OracleAQQueue` object. Multiple messages can be dequeued using the `DequeueArray` method.

See Also:
- "Dequeue" on page 12-64
- "DequeueArray" on page 12-66

Listen

Subscriber applications can use a `Listen` call to monitor multiple queues for subscriptions on different queues. This is a more scalable solution for cases where a subscriber application has subscribed to many queues and wishes to receive messages that arrive in any of the queues.

This is accomplished by calling the `Listen` method of the `OracleAQQueue` class, passing the list of subscriptions in form of an array.

See Also:  "Listen" on page 12-72

Notification

Subscriber applications can utilize the notification mechanism to get notifications about message availability in a queue. The applications can decide to skip or dequeue the message from the queue based on the information received.

A subscriber application must register for event notification on the queues from which it wants to receive notifications. This is represented by the `MessageAvailable` event on `OracleAQQueue`. The event is triggered when messages matching the subscriptions arrive.

Notifications can be registered as regular or grouping notifications. A time out value for these notifications can also be specified. Various notification options can be set using the `OracleAQQueue.Notification` property. Notifications set on an `OracleAQQueue` object gets cancelled automatically when the object gets disposed.

See Also:
- "MessageAvailable Event" on page 12-79
- "Notification" on page 12-61
Buffered messaging was introduced in Oracle Streams AQ 10g Release 2 (10.2). In buffered messaging, messages reside in a shared memory area. This makes it faster than persistent messaging. The messages are written to disk only when the total memory consumption of buffered messages approaches the available shared memory limit. Buffered messaging is ideal for applications that do not require the reliability and transaction support of Oracle Streams AQ persistent messaging.

Buffered and persistent messages use the same single-consumer or multi-consumer queues, and the same administrative and operational interfaces. They are distinguished from each other by a delivery mode parameter. When an application enqueues a message to an Oracle Streams AQ queue, it sets the delivery mode parameter as well.

The delivery mode parameter can be set on `OracleAQMessage` by modifying the `DeliveryMode` property. Buffered messaging is supported in all queue tables created with compatibility 8.1 or higher.

See Also: "DeliveryMode" on page 12-31

### Using ODP.NET for Advanced Queuing

.NET applications can use ODP.NET to access all the operational features of AQ such as Enqueuing, Dequeuing, Listen, and Notification.

Table 3–27 maps the AQ features to their corresponding ODP.NET implementation.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>ODP.NET Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Message</td>
<td>Create an <code>OracleAQMessage</code> object</td>
</tr>
<tr>
<td>Enqueue a single message</td>
<td>Specify the message as <code>OracleAQMessage</code>, queue as <code>OracleAQQueue</code> and enqueue options on <code>OracleAQQueue</code>, call <code>OracleAQQueue.Enqueue</code></td>
</tr>
<tr>
<td>Enqueue multiple messages</td>
<td>Specify the messages as an <code>OracleAQMessage</code> array in <code>OracleAQQueue.EnqueueArray</code></td>
</tr>
<tr>
<td>Dequeue a single message</td>
<td>Specify dequeue options on <code>OracleAQQueue</code> and call <code>OracleAQQueue.Dequeue</code></td>
</tr>
<tr>
<td>Dequeue multiple messages</td>
<td>Call <code>OracleAQQueue.DequeueArray</code></td>
</tr>
<tr>
<td>Listen for messages on Queue(s)</td>
<td>Call <code>OracleAQQueue.Listen</code>. To listen on multiple queues use static <code>Listen</code> method of <code>OracleAQQueue</code></td>
</tr>
<tr>
<td>Message Notifications</td>
<td>Use <code>OracleAQQueue.MessageNotification.Event</code> along with the <code>NotificationConsumers</code> property</td>
</tr>
</tbody>
</table>

Note: AQ samples are provided in the `ORACLE_BASE\ORACLE_HOME\ODP.NET\Samples` directory.

### Enqueuing and Dequeuing Example

The following example demonstrates enqueuing and dequeuing messages using a single consumer queue. The first part of the example performs the requisite database setup for the database user, SCOTT. The second part of the example demonstrates enqueuing and dequeuing messages.

-- Part I: Database setup required for this demo
-- SQL to grant appropriate privilege to database user, SCOTT

SQL> ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY Pwd4Sct;
User altered.
SQL> GRANT ALL ON DBMS_AQADM TO scott;

-- PL/SQL to create queue-table and queue and start queue for SCOTT

BEGIN
    DBMS_AQADM.CREATE_QUEUE_TABLE(
        queue_table=>'scott.test_q_tab',
        queue_payload_type=>'RAW',
        multiple_consumers=>FALSE);
    DBMS_AQADM.CREATE_QUEUE(
        queue_name=>'scott.test_q',
        queue_table=>'scott.test_q_tab');
    DBMS_AQADM.START_QUEUE(queue_name=>'scott.test_q');
END;
/

-- PL/SQL to stop queue and drop queue & queue-table from SCOTT

BEGIN
    DBMS_AQADM.STOP_QUEUE('scott.test_q');
    DBMS_AQADM.DROP_QUEUE(
        queue_name => 'scott.test_q',
        auto_commit => TRUE);
    DBMS_AQADM.DROP_QUEUE_TABLE(
        queue_table => 'scott.test_q_tab',
        force => FALSE,
        auto_commit => TRUE);
END;
/
-- End of Part I, database setup.

//Part II: Enqueuing and dequeuing messages
//C#
using System;
using System.Text;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

namespace ODPSample
{
    /// <summary>
    /// Demonstrates Enqueuing and Dequeuing raw message using a single consumer queue
    /// </summary>
    class EnqueueDequeue
    {
        static void Main(string[] args)
        {
            // Create connection
        }
}
```
string constr = "user id=scott;password=Pwd4Sct;data source=oracle";
OracleConnection con = new OracleConnection(constr);

// Create queue
OracleAQQueue queue = new OracleAQQueue("scott.test_q", con);

try {
    // Open connection
    con.Open();

    // Begin txn for enqueue
    OracleTransaction txn = con.BeginTransaction();

    // Set message type for the queue
    queue.MessageType = OracleAQMessageType.Raw;

    // Prepare message and RAW payload
    OracleAQMessage enqMsg = new OracleAQMessage();
    byte[] bytePayload = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    enqMsg.Payload = bytePayload;

    // Prepare to Enqueue
    queue.EnqueueOptions.Visibility = OracleAQVisibilityMode.OnCommit;

    // Enqueue message
    queue.Enqueue(enqMsg);

    Console.WriteLine("Enqueued Message Payload      : "+ByteArrayToString(enqMsg.Payload as byte[]));
    Console.WriteLine("MessageId of Enqueued Message : "+ByteArrayToString(enqMsg.MessageId));

    // Enqueue txn commit
    txn.Commit();

    // Begin txn for Dequeue
    txn = con.BeginTransaction();

    // Prepare to Dequeue
    queue.DequeueOptions.Visibility = OracleAQVisibilityMode.OnCommit;
    queue.DequeueOptions.Wait = 10;

    // Dequeue message
    OracleAQMessage deqMsg = queue.Dequeue();

    Console.WriteLine("Dequeued Message Payload      : "+ByteArrayToString(deqMsg.Payload as byte[]));
    Console.WriteLine("MessageId of Dequeued Message : "+ByteArrayToString(deqMsg.MessageId));

    // Dequeue txn commit
   txn.Commit();
}
catch (Exception e) {
    Console.WriteLine("Error: {0}", e.Message);
}
finally {
```
Database Change Notification Support

Oracle Data Provider for .NET provides a notification framework that supports Continuous Query Notification, enabling applications to receive client-side notifications when there is a change in a query result set, schema objects, or the state of the database, even if no Oracle Data Provider for .NET database connection exists. Using Continuous Query Notification, an application can maintain the validity of the client-side cache (for example, the ADO.NET `DataSet`) easily.

---

**Note:** The ODP.NET Database Change Notification feature uses the Continuous Query Notification feature in the Oracle database.

---

**Note:** Database change notification is not supported in a .NET stored procedure.

---

Using the notification framework, applications can specify a query result set as a registered query for notification request on the database, and create this notification registration to maintain the validity of the query result set. When there is a change on the database that could affect the client-side cache's query results, the notification framework notifies the application.

---

**Note:** The content of a change notification is referred to as an invalidation message. It indicates that the query result set is now invalid and provides information about the changes.

---

Based on the information provided by the invalidation message, the application can then act accordingly. For example, the application might need to refresh its own copy of the data for the registered query that is stored locally in the application.
By default, Windows Vista and Windows XP Service Pack 2 and later enable the Windows Firewall to block virtually all TCP network ports to incoming connections. Therefore, for Continuous Query Notification to work properly on these operating systems, the Windows Firewall must be configured properly to allow specific executables to open specific ports.

See Also: Oracle Database Advanced Application Developer’s Guide for further information on Continuous Query Notification

Beginning with Oracle Database 11g and ODP.NET 11g (11.1), Database Change Notification queries can be query-based (default) or object-based. The query-based registrations allow ODP.NET to notify applications when the selected rows have changed in the database. The object-based registrations allow ODP.NET to notify applications for any changes that occur in the table(s) containing the selected rows.

Query-based registrations have two modes: guaranteed mode and best-effort mode. In guaranteed mode, any database change notification ensures that a change occurred to something contained in the queried result set. However, if a query is complex, then it cannot be registered in guaranteed mode. Best-effort mode is used in such cases.

Best-effort mode simplifies the query for query-based registration. No notifications are lost from the simplification. However, the simplification may cause false positives, as the simpler version’s query result could change when the original query result would not.

There still remain some restrictions on which queries can have best-effort mode query-based registrations. In such cases, developers can use object-based registrations, which can register most query types. Object-based registrations generate notifications when the query object changes, even if the actual query result does not. This also means that object-based registrations are more prone to false positives than query-based registrations. Developers should be aware of the relative strengths and weaknesses of each database change notification option and choose the one that best suits their requirements.

See Also:
- "Configuring a Port to Listen for Database Notifications” on page 2-13
- Oracle Database Advanced Application Developer’s Guide for further details on the requirements for Continuous Query Notification

This section contains the following topics:
- Database Change Notification Classes
- Supported Operations
- Requirements of Notification Registration
- Using Database Change Notification
Database Change Notification Classes

The following classes are associated with Continuous Query Notification Support:

- **OracleDependency**
  
  Represents a dependency between an application and an Oracle database based on the database events which the application is interested in. It contains information about the dependency and provides the mechanism to notify the application when specified database events occurs. The OracleDependency class is also responsible for creating the notification listener to listen for database notifications. There is only one database notification listener for each application domain. This notification listener terminates when the application process terminates.

  The dependency between the application and the database is not established when the OracleDependency object is created. The dependency is established when the command that is associated with this OracleDependency object is executed. That command execution creates a database change notification registration in the database.

  When a change has occurred in the database, the HasChanges property of the OracleDependency object is set to true. Furthermore, if an event handler was registered with the OnChange event of the OracleDependency object, the registered event handler function will be invoked.

- **OracleNotificationRequest**
  
  Represents a notification request to be registered in the database. It contains information about the request and the properties of the notification.

- **OracleNotificationEventArgs**
  
  Represents the invalidation message generated for a notification when a specified database event occurs and contains details about that database event.

See Also: 
- "OracleDependency Class" on page 9-2
- "OracleNotificationRequest Class" on page 9-22
- "OracleNotificationEventArgs Class" on page 9-30

Supported Operations

The ODP.NET notification framework in conjunction with Continuous Query Notification supports the following activities:

- Creating a notification registration by:
  - Creating an OracleDependency instance and binding it to an OracleCommand instance.

- Grouping multiple notification requests into one registration by:
  - Using the OracleDependency.AddCommandDependency method.
  - Setting the OracleCommand.Notification request using the same OracleNotificationRequest instance.

- Registering for Database Change Notification by:
– Executing the OracleCommand. If either the notification property is null or NotificationAutoEnlist is false, the notification will not be made.

■ Removing notification registration by:
– Using the OracleDependency.RemoveRegistration method.
– Setting the Timeout property in the OracleNotificationRequest instance before the registration is created.
– Setting the IsNotifiedOnce property to true in the OracleNotificationRequest instance before the registration is created. The registration is removed once a database notification is sent.

■ Ensuring Change Notification Persistence by:
– Specifying whether or not the invalidation message is queued persistently in the database before delivery. If an invalidation message is to be stored persistently in the database, then the change notification is guaranteed to be sent. If an invalidation message is stored in an in-memory queue, the change notification can be received faster, however, it could be lost upon database shutdown or crashes.

■ Retrieving notification information including:
– The changed object name.
– The schema name of the changed object.
– Database events that cause the notification, such as insert, delete, and so on.
– The RowID of the modified object row.

In Oracle SQL, the ROWIDTOCHAR(ROWID) and ROWIDTONCHAR(ROWID) functions convert a ROWID value to VARCHAR2 and NVARCHAR data types, respectively.

If these functions are used within a SQL statement, ROWIDs are not returned in the OracleNotificationEventArgs object that is passed to the database change notification callback.

■ Defining the listener port number.

By default, the static OracleDependency.Port property is set to -1. This indicates that the ODP.NET listens on a port that is randomly picked when ODP.NET registers a database change notification request for the first time during the execution of an application.

ODP.NET creates only one listener that listens on one port within an application domain. Once ODP.NET starts the listener, the port number cannot be changed; Changes to the static OracleDependency.Port property will generate an error if a listener has already been created.

See Also:

■ "OracleCommand Class" on page 5-2
■ "Notification" on page 5-21
■ "NotificationAutoEnlist" on page 5-22
■ "OracleDependency Class" on page 9-2
■ "OracleNotificationEventArgs Class" on page 9-30
Requirements of Notification Registration

The connected user must have the CHANGE NOTIFICATION privilege to create a notification registration.

This SQL statement grants the CHANGE NOTIFICATION privilege:

```
grant change notification to user name
```

This SQL statement revokes the CHANGE NOTIFICATION privilege:

```
revoke change notification from user name
```

Using Database Change Notification

This section describes what the application should do, and the flow of the process, when an application uses Continuous Query Notification to receive notifications for any changes in the registered query result set.

Application Steps

The application should do the following:

1. Create an OracleDependency instance.
2. Assign an event handler to the OracleDependency.OnChange event property if the application wishes to have an event handler invoked when database changes are detected. Otherwise, the application can choose to poll on the HasChanges property of the OracleDependency object. This event handler is invoked when the change notification is received.
3. Set the port number for the listener to listen on. The application can specify the port number for one notification listener to listen on. If the application does not specify a port number, a random one is used by the listener.
4. Bind the OracleDependency instance to an OracleCommand instance that contains the actual query to be executed. Internally, the Continuous Query Notification request (an OracleNotificationRequest instance) is created and assigned to the OracleCommand.Notification property.

Flow of Notification Process

1. When the command associated with the notification request is executed, the notification registration is created in the database. The command execution must return a result set, or contain one or more REF cursors for a PL/SQL stored procedure.
2. ODP.NET starts the application listener on the first successful notification registration.
3. When a change related to the registration occurs in the database, the application is notified through the event delegate assigned to the OracleDependency.OnChange event property, or the application can poll the OracleDependency.HasChanges property.

The following example demonstrates the database change notification feature.

```c
// Database Setup
// NOTE: unless the following SQL command is executed,
// ORA-29972 will be obtained from running this sample
/*
grant change notification to scott;
```
using System;
using System.Threading;
using System.Data;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;

namespace NotificationSample
{
    public class MyNotificationSample
    {
        public static bool IsNotified = false;

        public static void Main(string[] args)
        {
            // To Run this sample, make sure that the change notification privilege
            // is granted to scott.
            string constr = "User Id=scott;Password=tiger;Data Source=oracle;"
            OracleConnection con = null;
            OracleDependency dep = null;

            try
            {
                con = new OracleConnection(constr);
                OracleCommand cmd = new OracleCommand("select * from emp", con);
                con.Open();

                // Set the port number for the listener to listen for the notification
                // request
                OracleDependency.Port = 1005;

                // Create an OracleDependency instance and bind it to an OracleCommand
                // instance.
                // When an OracleDependency instance is bound to an OracleCommand
                // instance, an OracleNotificationRequest is created and is set in the
                // OracleCommand's Notification property. This indicates subsequent
                // execution of command will register the notification.
                // By default, the notification request is using the Database Change
                // Notification.
                dep = new OracleDependency(cmd);

                // Add the event handler to handle the notification. The
                // OnMyNotification method will be invoked when a notification message
                // is received from the database
                dep.OnChange +=
                    new OnChangeEventHandler(MyNotificationSample.OnMyNotification);

                // The notification registration is created and the query result sets
                // associated with the command can be invalidated when there is a
                // change. When the first notification registration occurs, the
                // notification listener is started and the listener port number
                // will be 1005.
                cmd.ExecuteNonQuery();

                // Updating emp table so that a notification can be received when
                // the emp table is updated.
            }
        }
    }
}
// Start a transaction to update emp table
OracleTransaction txn = con.BeginTransaction();
// Create a new command which will update emp table
string updateCmdText =
    "update emp set sal = sal + 10 where empno = 7782";
OracleCommand updateCmd = new OracleCommand(updateCmdText, con);
// Update the emp table
updateCmd.ExecuteNonQuery();
//When the transaction is committed, a notification will be sent from
//the database
txn.Commit();
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}

con.Close();
// Loop while waiting for notification
while(MyNotificationSample.IsNotified == false)
{
    Thread.Sleep(100);
}

public static void OnMyNotificaton(object src,
    OracleNotificationEventArgs arg)
{
    Console.WriteLine("Notification Received");
    DataTable changeDetails = arg.Details;
    Console.WriteLine("Data has changed in {0}",
        changeDetails.Rows[0]["ResourceName"]);
    MyNotificationSample.IsNotified = true;
}

Best Practice Guidelines and Performance Considerations

This section provides guidelines for working with Continuous Query Notification and the ODP.NET notification framework, and discusses the performance impacts.

Every change notification registration consumes database memory, storage or network resources, or some combination thereof. The resource consumption further depends on the volume and size of the invalidation message. In order to scale well with a large number of mid-tier clients, Oracle recommends that the client implement these best practices:

- Few and mostly read-only tables

  There should be few registered objects, and these should be mostly read-only, with very infrequent invalidations. If an object is extremely volatile, then a large number of invalidation notifications are sent, potentially requiring a lot of space (in memory or on disk) in the invalidation queue. This is also true if a large number of objects are registered.

- Few rows updated for each table

  Transactions should update (or insert or delete) only a small number of rows within the registered tables. Depending on database resources, a whole table could
be invalidated if too many rows are updated within a single transaction, for a
given table.
This policy helps to contain the size of a single invalidation message, and reduces
disk storage for the invalidation queue.

See Also: Oracle Database Advanced Application Developer’s Guide for
further information on Continuous Query Notification

OracleDataAdapter Safe Type Mapping

The ODP.NET OracleDataAdapter class provides the Safe Type Mapping feature to
ensure that the following Oracle data types do not lose data when converted to their
closely related .NET types in the DataSet:

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

This section includes the following topics:

- Comparison Between Oracle Data Types and .NET Types
- SafeMapping Property

Comparison Between Oracle Data Types and .NET Types

The following sections provide more details about the differences between the Oracle
data types and the corresponding .NET types. In general, the Oracle data types allow a
greater degree of precision than the .NET types do.

Oracle NUMBER Type to .NET Decimal Type

The Oracle data type NUMBER can hold up to 38 precision, and the .NET Decimal
type can hold up to 28 precision. If a NUMBER data type that has more than 28 precision is
retrieved into a .NET Decimal type, it loses precision.

Table 3–28 lists the maximum and minimum values for Oracle NUMBER and .NET
Decimal types.

<table>
<thead>
<tr>
<th>Value Limits</th>
<th>Oracle NUMBER</th>
<th>.NET Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>9,99999999999999999999999999999999999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>79,228,162,514,264,337,593,543,950,335</td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td>-9,999999999999999999999999999999999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-79,228,162,514,264,337,593,543,950,335</td>
<td></td>
</tr>
</tbody>
</table>

Oracle Date Type to .NET DateTime Type

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

Table 3–29 lists the maximum and minimum values for Oracle Date and .NET
DateTime types.
Oracle TimeStamp Type to .NET DateTime Type

Similar to the DATE data type, the Oracle TimeStamp data type can represent a date in BC, and a .NET DateTime type cannot. If a TimeStamp that goes to BC is retrieved into a .NET DateTime type, it loses data. The Oracle TimeStamp type can represent values in units of e-9; the .NET DateTime type can represent only values in units of e-7. The Oracle TimeStamp with time zone data type can store time zone information, and the .NET DateTime type cannot.

Table 3–30 lists the maximum and minimum values for Oracle TimeStamp and .NET DateTime types.

<table>
<thead>
<tr>
<th>Value Limits</th>
<th>Oracle TimeStamp</th>
<th>.NET DateTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>Jan 1, 4712 BC 00:00:00.000000000</td>
<td>Jan 1, 0001 AD 00:00:00.000000000</td>
</tr>
</tbody>
</table>

Oracle INTERVAL DAY TO SECOND to .NET TimeSpan

The Oracle data type INTERVAL DAY TO SECOND can hold up to 9 precision, and the .NET TimeSpan type can hold up to 7 precision. If an INTERVAL DAY TO SECOND data type that has more than 7 precision is retrieved into a .NET TimeSpan type, it loses precision. The Oracle INTERVAL DAY TO SECOND type can represent values in units of e-9, and the .NET TimeSpan type can represent only values in units of e-7.

Table 3–31 lists the maximum and minimum values for Oracle INTERVAL DAY TO SECOND and .NET DateTime types.

<table>
<thead>
<tr>
<th>Value Limits</th>
<th>Oracle INTERVAL DAY TO SECOND</th>
<th>.NET TimeSpan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>+9999999999 23:59:59.999999999</td>
<td>+10675199 02:48:05.4775807</td>
</tr>
</tbody>
</table>

SafeMapping Property

The OracleDataAdapter Safe Type Mapping feature prevents data loss when populating Oracle data for any of these types into a .NET DataSet. 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

By default, Safe Type Mapping is disabled.
Using Safe Type Mapping

To use the Safe Type Mapping feature, the `OracleDataAdapter.SafeMapping` property must be set with a hash table 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. 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 hash table that correspond to invalid column names are ignored.

---

Safe Type Mapping as a string is more readable without further conversion. Converting certain Oracle data types 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.

**SafeTyping Example**

```csharp
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
        // 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:  "SafeMapping" on page 5-127

OracleDataAdapter Requery Property

The OracleDataAdapter Requery property controls whether or not 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, reexecuting the query may not be desirable.

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

When the Requery property is set to false, the DataSet has all the 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 supported only 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 an OracleRefCursor object, 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 call and the OracleRefCursor object must be available for the subsequent Fill calls.
Guaranteeing Uniqueness in Updating DataSet to Database

This section describes how the OracleDataAdapter object configures the PrimaryKey and Constraints properties of the DataTable object which guarantee uniqueness when the OracleCommandBuilder object 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 object with the database.

In this process, the OracleCommandBuilder object 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 object change affects only a single row, there must be a set of DataColumn objects in the DataTable for which all rows in the DataTable have a unique set of values. The set of DataColumn objects indicated by the properties DataTable.PrimaryKey and DataTable.Constraints meets this requirement.

The OracleCommandBuilder object determines uniqueness in the DataTable by checking if the DataTable.PrimaryKey is not a null value or if there exists a UniqueConstraint object in the DataTable.Constraints collection.

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

This section includes the following topics:
- What Constitutes Uniqueness in DataRow Objects?
- Configuring PrimaryKey and Constraints Properties
- Updating Without PrimaryKey and Constraints Configuration

What Constitutes Uniqueness in DataRow Objects?

This section describes the minimal conditions that must be met to guarantee uniqueness of DataRow objects. 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 object if any one of the following is true:
- All the columns of the primary key are in the select list of the OracleDataAdapter.SelectCommand property.
- All the columns of a unique constraint are in the select list of the OracleDataAdapter.SelectCommand property, 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 property, with at least one of the involved columns having a NOT NULL constraint defined on it.

See Also:
- "Requery" on page 5-127
- "SelectCommand" on page 5-128
Guaranteeing Uniqueness in Updating DataSet to Database

- A ROWID is present in the select list of the OracleDataAdapter.SelectCommand property.

---

**Note:** A set of columns, on which a unique constraint has been defined or a unique index has been created, requires at least one column that cannot be null for the following reason: if all the columns of the column set can be null, then multiple rows could exist that 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 DataRow Objects?" on page 3-131 are met, then the DataTable.PrimaryKey or DataTable.Constraints properties can be set.

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

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 property.
2. If a set of columns that meets the following criteria is returned in the select list, it is set as the DataTable.PrimaryKey property.
   
   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 object is added to the DataTable.Constraints collection, but the DataTable.PrimaryKey property 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 property.

Additionally, the OracleDataAdapter.FillSchema method performs as follows:

- Setting the DataTable.PrimaryKey property implicitly creates a UniqueConstraint object.
- If a column is part of the DataTable.PrimaryKey property or the UniqueConstraint object, or both, it will be repeated for each occurrence of the column in the select list.

---

**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 the OracleDataAdapter.FillSchema method, the OracleCommandBuilder object
directly checks the select list of the `OracleDataAdapter.SelectCommand` property to determine if it guarantees uniqueness in the `DataTable`. However this check results in a database round-trip to retrieve the metadata for the `SELECT` statement of the `OracleDataAdapter.SelectCommand`.

Note that `OracleCommandBuilder` object 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 10-2

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
- 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. These settings can be obtained by calling the `OracleGlobalization.GetClientInfo` static method.

The following example retrieves the client globalization settings:

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

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

        Console.WriteLine("Client machine language: " + ClientGlob.Language);
        Console.WriteLine("Client characterset: " + 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 be obtained only after establishing a connection against the database. The session globalization settings can be obtained by calling the `GetSessionInfo` method on the `OracleConnection` object. Invoking this method returns an instance of an `OracleGlobalization` class 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 (NLS)) registry settings. The session settings can be updated 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` class. The `SetThreadInfo` static method of the `OracleGlobalization` class 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 the System.Threading.Thread.CurrentCulture property do not impact the OracleGlobalization settings of the thread or the session, or the reverse.

The following example shows how the thread’s globalization settings are used by the ODP.NET Types:

```csharp
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 OracleGlobalization settings of the client computer. The client character set of the local computer 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 that 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. Specific thread globalization settings are used in most cases, depending on the ODP.NET type, by the following:

- The **ToString** method
- The **Parse** static method
- Constructors that accept .NET string data
- Conversion operators to and from .NET strings

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 on the ODP.NET type methods that convert between ODP.NET types and .NET string types, and to identify which thread globalization settings are used for that particular method, read the remarks in **Chapter 10**.

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 is not affected by thread settings because 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.

*See Also:*

- "OracleDataAdapter Safe Type Mapping" on page 3-127
- Chapter 10, "Oracle Data Provider for .NET Globalization Classes"
- Chapter 14, "Oracle Data Provider for .NET Types Structures"

**Operations Sensitive to Session Globalization Parameters**

Session globalization settings affect any data that is retrieved from or sent to the database 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 property 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 SQL statements modified by the provider
- Connection pooling statistics such as enlistment and delistment
- Thread ID (entry and exit)
- HA Events and Load Balancing information
- Distributed Transactions
- Self-tuning information
- User-mode dumps upon unmanaged exceptions

To enable ODP.NET for tracing, `TraceFileName`, `TraceLevel`, and `TraceOption` must be set appropriately either in the Windows Registry or in an XML configuration file.

See Also: "Configuring Oracle Data Provider for .NET" on page 2-5 for further details
This chapter discusses server-side features provided by Oracle Data Provider for .NET.

With the support for .NET stored procedures in Oracle Databases for Windows that Oracle Database Extensions for .NET provides, ODP.NET can be used to access Oracle data through the implicit database connection that is available from the context of the .NET stored procedure execution. Explicit user connections can also be created to establish connections to the database that hosts the .NET stored procedure or to other Oracle Databases.

See Also: Oracle Database Extensions for .NET Developer’s Guide

This chapter contains these topics:

- Introducing .NET Stored Procedure Execution Using ODP.NET
- Limitations and Restrictions on ODP.NET Within .NET Stored Procedure
- Porting Client Application to .NET Stored Procedure

Introducing .NET Stored Procedure Execution Using ODP.NET

Oracle Data Provider for .NET classes and APIs provide data access to the Oracle Database from a .NET client application and from .NET stored procedures and functions.

However, some limitations and restrictions exist when Oracle Data Provider for .NET is used within a .NET stored procedure. These are discussed in the next section.

The following is a simple .NET stored procedure example.

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

public class CLRLibrary1
{
    // .NET Stored Function returning the DEPTNO of the employee whose
    // EMPNO is 'empno'
    public static uint GetDeptNo(uint empno)
    {
        uint deptno = 0;
    }
}
```
Limitations and Restrictions on ODP.NET Within .NET Stored Procedure

This section covers important concepts that apply when Oracle Data Provider for .NET is used within a .NET stored procedure.

Implicit Database Connection

Within a .NET stored procedure, an implicit database connection is available for use to access Oracle data. This implicit database connection should be used rather than establishing a user connection because the implicit database connection is already established by the caller of the .NET stored procedure, thereby minimizing resource usage.

To obtain an OracleConnection object in a .NET stored procedure that represents the implicit database connection, set the ConnectionString property of the OracleConnection object to "context connection=true" and invoke the Open method. No connection string attributes can be used with "context connection=true", except the Statement Cache Size attribute.

The availability of the implicit database connection can be checked at run time through the static OracleConnection.IsAvailable property. This property always returns
true when Oracle Data Provider for .NET is used within a .NET stored procedure. Otherwise, false is returned.

**Note:** DBLinks are not supported in .NET stored procedures.

Only one implicit database connection is available within a .NET stored procedure invocation. To establish more connections in addition to the implicit database connection, an explicit connection must be created. When the `Close` method is invoked on the `OracleConnection` that represents the implicit database connection, the connection is not actually closed. Therefore, the `Open` method of the same or another `OracleConnection` object can be invoked to obtain the connection that represents the implicit database connection.

The implicit database connection can only be acquired by the `Open` method invocation by a native Oracle thread that initially invokes the .NET stored procedure. However, threads spawned from the native Oracle thread can use implicit database connections that are obtained by the native Oracle thread.

**See Also:** "IsAvailable" on page 5-71

### Transaction Support

The .NET stored procedure execution automatically inherits the current transaction on the implicit database connection. No explicit transaction can be started, committed, or rolled back inside a .NET stored procedure on a Context connection. However, explicit transaction can be started, committed, or rolled back inside a .NET stored procedure on a Client connection.

For example, `OracleConnection.BeginTransaction` is not allowed inside a .NET stored procedure for a context connection, but is allowed for a client connection. .NET stored procedures do not support distributed transactions. If you have enlisted a client connection in a distributed transaction and call a .NET stored procedure or function, an error occurs.

If a .NET stored procedure or function performs operations on the database that are required to be part of a transaction, the transaction must be started prior to calling the .NET stored procedure. Any desired commit or rollback must be performed after returning from the .NET stored procedure or function.

The following example consists of a client application and a .NET stored procedure, `InsertRecordSP`, that inserts an employee record into an `EMP` table.

**Example (.NET Stored Procedure)**

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

// This class represents an Oracle .NET stored procedure that inserts
// an employee record into an EMP table of SCOTT schema.
public class InsertRecordSP
{
    // This procedure will insert a row into the emp database
    // For simplicity we are using only two parameters, the rest are hard coded
    public static void InsertRecord( int EmpNo, string EmpName )
    {
        if(OracleConnection.IsAvailable == true )
        {
            OracleConnection conn = new OracleConnection(
```

---

Oracle Data Provider for .NET Server-Side Features 4-3
Example (Client Application)
The example enters new employee, Bernstein, employee number 7950, into the EMP table.

```csharp
// C#
// This sample demonstrates how to start the transaction with ODP.NET client
// application and execute an Oracle .NET stored procedure that performs
// a DML operation. Since .NET stored procedure inherits the current
// transaction from the implicit database connection, DML operation
// in .NET stored procedure will not be in auto-committed mode.
// Therefore, it is up to the client application to do a COMMIT or ROLLBACK
// after returning from .NET stored procedure using System;
// using System.Data;
// using Oracle.DataAccess.Client;
// In this class we are starting a transaction on the client side and
// executing a .NET stored procedure, which inserts a record into EMP
// table and then verifies record count before and after COMMIT statement
class TransactionSample
{
    static void Main(string[] args)
    {
        OracleConnection Conn = null;
        OracleTransaction Txn = null;
        OracleCommand Cmd = null;
        try
        {
            Console.WriteLine( "Sample: Open DB connection in non auto-committed " + "'mode,' + "DML operation performed by .NET stored " + "procedure doesn't have an effect before COMMIT " + "is called.'");
            // Create and Open oracle connection
            Conn = new OracleConnection();
            Conn.ConnectionString = "User Id=scott;Password=tiger;Data Source=oracle;";
            Conn.Open();
            // Start transaction
            Txn = Conn.BeginTransaction( IsolationLevel.ReadCommitted );
            // Create command object
            Cmd = new OracleCommand();
            Cmd.Connection = Conn;
            Cmd.CommandType = CommandType.StoredProcedure;
```
Cmd.CommandText = "InsertRecord"; // .NET Stored procedure
// Parameter settings
OracleParameter EmpNoPrm = Cmd.Parameters.Add("empno", OracleDbType.Int32);
EmpNoPrm.Direction = ParameterDirection.Input;
EmpNoPrm.Value = 7950;
OracleParameter EmpNamePrm = Cmd.Parameters.Add("ename", OracleDbType.Varchar2, 10);
EmpNamePrm.Direction = ParameterDirection.Input;
EmpNamePrm.Value = "Bernstein";
// Execute .NET stored procedure
Cmd.ExecuteNonQuery();
Console.WriteLine( "Number of record(s) before COMMIT {0}",
    RecordCount() );
Txn.Commit();
Console.WriteLine( "Number of record(s) after COMMIT {0}",
    RecordCount() );
}
catch( OracleException OE )
{
    Console.WriteLine( OE.Message );
}
finally
{
    // Cleanup objects
    if( null != Txn )
        Txn.Dispose();
    if( null != Cmd )
        Cmd.Dispose();
    if( null != Conn && Conn.State == ConnectionState.Open )
        Conn.Close();
}
}
static int RecordCount()
{
    int EmpCount = 0;
    OracleConnection Conn = null;
    OracleCommand Cmd = null;
    try
    {
        Conn = new OracleConnection("User Id=scott;Password=tiger;" +
            "Data Source=oracle;" );
        Conn.Open();
        Cmd = new OracleCommand("SELECT COUNT(*) FROM EMP", Conn );
        Object o = Cmd.ExecuteScalar();
        EmpCount = Convert.ToInt32(o.ToString());
    }
catch( OracleException OE )
    {
        Console.WriteLine( OE.Message );
    }
finally
    {
        if( null != Cmd )
            Cmd.Dispose();
    }
    return EmpCount;
}
Unsupported SQL Commands

Transaction controls commands such as COMMIT, ROLLBACK, and SAVEPOINT are not supported in a .NET stored procedure.

Data definition commands such as CREATE and ALTER are not supported with an implicit database connection, but they are supported with an explicit user connection in a .NET stored procedure.

Oracle User-Defined Type (UDT) Support

UDTs are not supported within a context connection but they are supported with a client connection. UDTs are not supported as parameters to .NET stored procedures.

Porting Client Application to .NET Stored Procedure

All classes and class members provide the same functionality for both client applications and .NET stored procedures, unless it is otherwise stated.

Table 4–1 lists those classes or class members that have different behavior depending on whether or not they are used in a client application or in a .NET stored procedure.

Column Headings

The column heading for this table are:

Client application: The client application.

Implicit connection: The implicit database connections in a .NET stored procedure.

Explicit connection: The explicit user connections in a .NET stored procedure.

Table 4–1  API Support Comparison Between Client Application and .NET Stored Procedure

<table>
<thead>
<tr>
<th>Class or Class Members</th>
<th>Client Application</th>
<th>Implicit Connection/Explicit Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnChangeEventHandler Delegate</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-all members</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OracleDependency Class</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-all members</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OracleNotificationEventArgs Class</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-all members</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OracleNotificationRequest Class</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-all members</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OracleFailoverEventArgs Class</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-all members</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OracleFailoverEventHandler Delegate</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-all members</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OracleTransaction Class</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-all members</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OracleCommand Class</td>
<td>Yes</td>
<td>No #1/No #1</td>
</tr>
<tr>
<td>-Transaction Property</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4–1  (Cont.) API Support Comparison Between Client Application and .NET Stored Procedure

<table>
<thead>
<tr>
<th>Class or Class Members</th>
<th>Client Application</th>
<th>Implicit Connection/Explicit Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleConnection Class</td>
<td>Yes</td>
<td>Yes #3/Yes</td>
</tr>
<tr>
<td>-ConnectionTimeout Property</td>
<td>Yes</td>
<td>Yes #2/Yes</td>
</tr>
<tr>
<td>-DataSource Property</td>
<td>Yes</td>
<td>No/Yes</td>
</tr>
<tr>
<td>-BeginTransaction Method</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-ChangeDatabase Method</td>
<td>No</td>
<td>No/No</td>
</tr>
<tr>
<td>-Clone Method</td>
<td>Yes</td>
<td>No/Yes</td>
</tr>
<tr>
<td>-EnlistDistributedTransaction Method</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-OpenWithNewPassword Method</td>
<td>Yes</td>
<td>No/Yes</td>
</tr>
<tr>
<td>-Failover Event</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-OracleFailoverEventHandlerDelegate</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>ODP.NET Enumerations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-FailoverEvent Enumeration</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-FailoverReturnCode Enumeration</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-FailoverType Enumeration</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-OracleNotificationInfo Enumeration</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-OracleNotificationSource Enumeration</td>
<td>Yes</td>
<td>No/No</td>
</tr>
<tr>
<td>-OracleNotificationType Enumeration</td>
<td>Yes</td>
<td>No/No</td>
</tr>
</tbody>
</table>

Comments on Items in Table 4–1

1. Always returns null.
2. Implicit database connection always returns an empty string.
3. Implicit database connection always returns 0.
This chapter describes the following Oracle Data Provider for .NET classes.

- OracleCommand Class
- OracleCommandBuilder Class
- OracleConnection Class
- OracleDataAdapter Class
- OracleDatabase Class
- OracleDataReader Class
- OracleError Class
- OracleErrorCollection Class
- OracleException Class
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventHandler Delegate
- OracleParameter Class
- OracleParameterCollection Class
- OraclePermission Class
- OraclePermissionAttribute Class
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventHandler Delegate
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventHandler Delegate
- OracleTransaction Class
- OracleConnectionType Enumeration
- OracleCollectionType Enumeration
- OracleDBShutdownMode Enumeration
- OracleDBStartupMode 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
System.Object
    System.MarshalByRefObject
        System.ComponentModel.Component
            System.Data.Common.DbCommand (ADO.NET 2.0 only)

Declaration
// ADO.NET 2.0: C#
public sealed class OracleCommand : DbCommand, 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.

To minimize the number of open server cursors, OracleCommand objects should be explicitly disposed.

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);

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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 5–1.

Table 5–1 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
The OracleCommand static method is listed in Table 5–2.

Table 5–2 OracleCommand Static Method

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

OracleCommand Properties
OracleCommand properties are listed in Table 5–3.

Table 5–3 OracleCommand Properties

<table>
<thead>
<tr>
<th>Property</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 System.ComponentModel.Component</td>
</tr>
<tr>
<td>DesignTimeVisible</td>
<td>Specifies whether or not the OracleCommand object is visible on designer controls.</td>
</tr>
<tr>
<td>FetchSize</td>
<td>Specifies the size of OracleDataReader’s internal cache to store result set data</td>
</tr>
</tbody>
</table>
OracleCommand Public Methods

OracleCommand public methods are listed in Table 5–4.

Table 5–4  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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>CreateParameter</td>
<td>Creates a new instance of OracleParameter class</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
</tbody>
</table>
### Table 5–4  (Cont.) OracleCommand Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Prepare</td>
<td>This method is a no-op</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

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

**OracleCommand()**
This constructor instantiates a new instance of OracleCommand class.

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

**Remarks**
Default constructor.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members

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

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

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members

OracleCommand(string, OracleConnection)

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

Declaration

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

Parameters

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

- **OracleConnection**
  The connection to the Oracle database.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members
OracleCommand Static Methods

The `OracleCommand` static method is listed in Table 5–5.

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleCommand Class`
- `OracleCommand Members`
**OracleCommand Properties**

OracleCommand properties are listed in **Table 5–6**.

**Table 5–6  OracleCommand Properties**

<table>
<thead>
<tr>
<th>Property</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 System.ComponentModel.Component</td>
</tr>
<tr>
<td>DesignTimeVisible</td>
<td>Specifies whether or not the OracleCommand object is visible on designer controls.</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 of data that the OracleDataReader initially fetches for LOB columns</td>
</tr>
<tr>
<td>InitialLONGFetchSize</td>
<td>Specifies the amount of data that the OracleDataReader initially fetches for LONG and LONG RAW columns</td>
</tr>
<tr>
<td>Notification</td>
<td>Indicates that there is a notification request for the command</td>
</tr>
<tr>
<td>NotificationAutoEnlist</td>
<td>Indicates whether or not to register for a database change notification with the database automatically when the command is executed</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 cache to store one row of data</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Transaction</td>
<td>Specifies the OracleTransaction object in which the OracleCommand executes</td>
</tr>
<tr>
<td></td>
<td><em>Not supported in a .NET stored procedure</em></td>
</tr>
</tbody>
</table>
Table 5–6 (Cont.) OracleCommand Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdatedRowSource</td>
<td>Specifies how query command results are applied to the row being updated</td>
</tr>
<tr>
<td></td>
<td><em>Not supported in a .NET stored procedure</em></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 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 save changes to the database</td>
</tr>
</tbody>
</table>

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

AddRowid

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

Declaration

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

Property Value

bool

Remarks

Default is false.

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members
- "LOB Support" on page 3-83 for further information on how this property used with LOBs

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.

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**

// 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);

        if (cmd.AddToStatementCache)
        { Console.WriteLine("Added to the statement cache:" + cmd.CommandText); }
        else
        { Console.WriteLine("Not added to the statement cache:" + cmd.CommandText); }

        // 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;

        if (cmd.AddToStatementCache)
        { Console.WriteLine("Added to the statement cache:" + cmd.CommandText); }
        else
        { Console.WriteLine("Not added to the statement cache:" + cmd.CommandText); }
    }
}
Console.WriteLine("Not added to the statement cache:" + cmd.CommandText);

// 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 1-4
- OracleCommand Class
- OracleCommand Members
- "Statement Caching” on page 3-56
- ConnectionString on page 5-78

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 1-4
- OracleCommand Class
- OracleCommand Members
- "Array Binding" on page 3-53
- "Value" on page 5-277

**BindByName**

This property specifies the binding method in the collection.

**Declaration**

```csharp
// 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 1-4
- OracleCommand Class
- OracleCommand Members
- "Array Binding" on page 3-53
- "Value" on page 5-277

**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

// ADO.NET 2.0: C#
public override string CommandText {get; set;}

Property Value
A string.

Implements
IDbCommand

Remarks
The default is an empty string.

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

The effects of XmlCommandType values on CommandText are:

- XmlCommandType = None.
  CommandType property determines the contents of CommandText.
- XmlCommandType = Query.
  CommandText must be a SQL query. The SQL query should be a select statement. CommandType property is ignored.
- XmlCommandType property is Insert, Update, or Delete.
  CommandText must be an XML document. CommandType property is ignored.

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

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

Declaration

// ADO.NET 2.0: C#
public override int CommandTimeout {get; set;}

Property Value
int

Implements
IDbCommand.CommandTimeout

Exceptions
InvalidOperationException - 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. Other possible exceptions thrown after a command timeout expiration occurs include ORA-00936 and ORA-00604. 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 1-4
- OracleCommand Class
- OracleCommand Members
- [http://msdn.microsoft.com/library](http://msdn.microsoft.com/library) for detailed information about this Microsoft .NET Framework 1.1 feature

**CommandType**

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

Declaration

```csharp
// ADO.NET 2.0: C#
public override CommandType CommandType {get; 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 1-4
- 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 1-4
- OracleCommand Class
- OracleCommand Members

DesignTimeVisible

This property specifies whether or not the OracleCommand object is visible on designer controls.

Declaration

// C#
public override bool DesignTimeVisible { get; set; }

Property Value

A value that indicate whether or not OracleCommand object is visible in a control. The default is true.

Remarks

This property is used by developers to indicate whether or not OracleCommand object is visible in a control.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 = 131072.

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 database round-trip.

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

The RowSize and FetchSize properties handle UDT and XMLType data differently than other scalar data types. Because only a reference to the UDT and XMLType data is stored in the ODP.NET's internal cache, the RowSize property accounts for only the memory needed for the reference (which is very small) and not the actual size of the UDT and XMLType data. Thus, applications can inadvertently fetch a large number of UDT or XMLType instances from the database in a single database round-trip. This is because the actual size of UDT and XMLType data do not count against the FetchSize, and it would require numerous UDT and XMLType references to fill up the default cache size of 131072 bytes. Therefore, when fetching UDT or XMLType data, the FetchSize property must be appropriately configured to control the number of UDT and XMLType instances that are to be fetched, rather than the amount of the actual UDT and XMLType data to be fetched.

NOTE: For LOB and LONG data types, only the sizes specified in the InitialLOBFetchSize and InitialLONGFetchSize properties are accounted for by the RowSize property in addition to the metadata and reference information that is maintained by the cache for each LOB in the select list.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members
- OracleDataReader "FetchSize" on page 5-161

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

Declaration

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

InitialLONGFetchSize
**Property Value**

An `int` specifying the number of characters or bytes to fetch initially.

**Exceptions**

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

**Remarks**

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.

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

When `InitialLOBFetchSize` is set to `-1`, the entire LOB data is prefetched and stored in the fetch array. Calls to `GetString`, `GetChars` or `GetBytes` in `OracleDataReader` allow retrieving the entire data. In this case, the following methods are disabled:

- `GetOracleBlob`
- `GetOracleClob`
- `GetOracleClobForUpdate`
- `GetOracleBlobForUpdate`

This feature works for retrieving data from Oracle Database 9i release 2 (9.2) and later. The default value is `0`.

For Oracle Database 10g release 2 (10.2) and later:

The maximum value supported for `InitialLOBFetchSize` is 2 GB.

Prior to Oracle Database 10g release 2 (10.2), if the `InitialLOBFetchSize` is set to a nonzero value, `GetOracleBlob` and `GetOracleClob` methods were disabled. BLOB and CLOB data was fetched by using `GetBytes` and `GetChars` methods, respectively. In Oracle Database 10g release 2 (10.2), this restriction no longer exists. `GetOracleBlob` and `GetOracleClob` methods can be used for any `InitialLOBFetchSize` value zero or greater.

For releases prior to Oracle Database 10g release 2 (10.2):

The maximum value supported for `InitialLOBFetchSize` is 32 K.

To fetch more than the specified `InitialLOBFetchSize` value, 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)

If this property is set to 0, none of the preceding is required.
InitialLONGFetchSize

This property specifies the amount of data 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.

When InitialLONGFetchSize is set to -1, the entire LONG or LONG RAW data is prefetched and stored in the fetch array. Calls to GetString, GetChars, or GetBytes in OracleDataReader allow retrieving the entire data.

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 1-4
- OracleCommand Class
- OracleCommand Members
- "Obtaining LOB Data" on page 3-66
**Notification**

This instance property indicates that there is a notification request for the command.

**Declaration**

```csharp
// C#
public OracleNotificationRequest Notification {set; get;}
```

**Property Value**

A notification request for the command.

**Remarks**

When a changed notification is first registered, the client listener is started in order to receive any database notification. The listener uses the port number defined in the `OracleDependency.Port` static field. Subsequent change notification registrations use the same listener in the same client process and do not start another listener.

When `Notification` is set to an `OracleNotificationRequest` instance, a notification registration is created (if it has not already been created) when the command is executed. Once the registration is created, the properties of the `OracleNotificationRequest` instance cannot be modified. If the notification registration has already been created, the result set that is associated with the command is added to the existing registration.

When `Notification` is set to `null`, subsequent command executions do not require a notification request. If a notification request is not required, set the `Notification` property to `null`, or set the `NotificationAutoEnlist` property to `false`.

For Continuous Query Notification, a notification request can be used for multiple command executions. In that case, any query result set associated with different commands can be invalidated within the same registration.

When the `OracleDependency.OnChange` event is fired, if the `ROWID` column is explicitly included in the query (or `AddRowid` property is set to `true`), then the `Rowid` column contains `ROWID` values in the `DataTable` referenced by the `OracleNotificationEventArgs.Details` property. This behavior can be overridden by explicitly requesting for an inclusion and exclusion of `ROWID` values in the `OracleNotificationEventArgs` by setting the `OracleDependency.RowidInfo` to `OracleRowidInfo.Include` or `OracleRowidInfo.Exclude`, respectively.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleCommand Class`
- `OracleCommand Members`
- "Obtaining LONG and LONG RAW Data" on page 3-65 for further information
OracleCommand Properties

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members
- "Database Change Notification Support" on page 3-120
- Chapter 9, "Database Change Notification" on page 9-1

NotificationAutoEnlist

This instance property indicates whether or not to register for a database change notification with the database automatically when the command is executed.

Declaration

// C#
public bool NotificationAutoEnlist {set; get;}

Property Value

A bool value indicating whether or not to make a database change notification request automatically, when the command is executed. If NotificationAutoEnlist is set to true, and the Notification property is set appropriately, a database change notification request is registered automatically; otherwise, no database change notification registration is made.

Default value: true

Remarks

A notification request can be used for multiple command executions using the same OracleCommand instance. In that case, set the NotificationAutoEnlist property to true.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members
- "Database Change Notification Support" on page 3-120
- Chapter 9, "Database Change Notification" on page 9-1

Parameters

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

Declaration

// C#
public OracleParameterCollection Parameters {get;}

Property Value

OracleParameterCollection
Implements
IDbCommand

Remarks
Default value = an empty collection

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

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

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

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

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

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

Remarks
Default value = 0

The RowSize property is set to a nonzero value after the execution of a command that returns a result set. This property can be used at design time or dynamically during run time, to set the FetchSize, based on number of rows. For example, to enable the OracleDataReader to fetch N rows for each database round-trip, the OracleDataReader 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 1-4
- OracleCommand Class
- OracleCommand Members
- OracleDataReader "FetchSize" on page 5-17

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.

Remarks (.NET Stored Procedure)
Always returns null.

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

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

Declaration

// ADO.NET 2.0: C#
public override UpdateRowSource UpdatedRowSource {get; set;}

Property Value
An UpdateRowSource.

Implements
IDbCommand

Exceptions
ArgumentException - The UpdateRowSource value specified is invalid.

Remarks
Always returns UpdateRowSource,

Set accessor throws an ArgumentException if the value is other than UpdateRowSource.None.
XmlCommandType
This property specifies the type of XML operation on the OracleCommand.

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

Property Value
An OracleXmlCommandType.

Remarks
Default value is None.

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

See Also:
■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleCommand Class
■ OracleCommand Members

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

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

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

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

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

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
- OracleCommand Class
- OracleCommand Members
OracleCommand Public Methods

OracleCommand public methods are listed in Table 5–7.

Table 5–7  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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>CreateParameter</td>
<td>Creates a new instance of OracleParameter class</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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>Execute 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 System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Prepare</td>
<td>This method is a no-op</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

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

Cancel

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

// ADO.NET 2.0: C#
public override void Cancel();

Implements

IDbCommand.Cancel

Remarks

If cancellation of the command succeeds, an exception is thrown. If 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. Invoking the Cancel method does not guarantee that the command executing at the time will always be cancelled. The execution may complete before it can be terminated. In such cases, no exception is thrown.

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 command designated for cancellation may complete before the Cancel invocation is effective. If this happens, a command executed by a different OracleCommand could be cancelled instead.

There are several ways to avoid this non-deterministic situation that the Cancel method can cause:

- The application can create just one OracleCommand object for each connection. Doing so assures that the Cancel invocation only cancels commands executed by the OracleCommand object using a particular connection.
- Command executions in the application are synchronized between OracleCommand objects that use the same connection.

These suggestions do not apply if Cancel is not used in the application.

Because 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, the transaction can be rolled back for data integrity. Other possible exceptions thrown after a command cancellation occurs include ORA-00936 and ORA-00604. 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 same 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());
}
Console.WriteLine("Cancel done.");
Console.WriteLine("CancelEvent.Set()"鲣;
CancelEvent.Set();
}

// Execution of the command with a potential of cancelling
public void ExecuteWithinLimitedTime()
{
    for (int i = 0; i < 5; i++)
    {
        Monitor.Enter(typeof(CancellationToken));
        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(CancellationToken));
        }
    }
}

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 1-4
- 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 1-4
- 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 1-4
- 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

// ADO.NET 2.0: C#
public override 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";
        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 later, to be installed in the database. Oracle XDK can be downloaded from Oracle Technology Network (OTN).

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members
- http://otn.oracle.com/

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 1-4
- OracleCommand Class
- OracleCommand Members

ExecuteReader()

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

Declaration

// C#
public OracleDataReader ExecuteReader();

Return Value

An OracleDataReader.

Implements

IDbCommand

Exceptions

InvalidOperationException - The command cannot be executed.

Remarks

When the CommandType property is set to CommandType.StoredProcedure, the CommandText property should be set to the name of the stored procedure.
The specified command executes this stored procedure when `ExecuteReader` is called. If parameters for the stored procedure consist of REF CURSOR objects, behavior differs depending on whether `ExecuteReader()` or `ExecuteNonQuery()` is called. If `ExecuteReader()` is invoked, REF CURSOR objects can be accessed through the `OracleDataReader` that is returned.

If more than one REF CURSOR is returned from a single execution, subsequent REF CURSOR objects can be accessed sequentially by the `NextResult` method on the `OracleDataReader`. If the `ExecuteNonQuery` method is invoked, the output parameter value can be cast to an `OracleRefCursor` type and the `OracleRefCursor` object then can be used to either populate a `DataSet` or create an `OracleDataReader` object from it. This approach provides random access to all the REF CURSOR objects returned as output parameters.

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 3-65.

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();
    }
}
```
ExecuteReader(CommandBehavior)

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

Declaration

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

Parameters

- behavior
  The 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 command 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 OracleRefCursor for more details. See "OracleRefCursor Class" on page 13-113.

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 3-65.

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 1-4
- OracleCommand Class
- OracleCommand Members
- "OracleRefCursor Class" on page 13-113

**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
// ADO.NET 2.0: C#
public override 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();
    }
}
```
```csharp
Console.WriteLine("There are {0} rows in table emp", count);

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

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members
- Oracle XML DB Developer’s Guide
- http://otn.oracle.com/
ExecuteXmlReader

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

Declaration

// C#
public XmlReader ExecuteXmlReader();

Return Value

An XmlReader.

Remarks

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

- XmlCommandType = OracleXmlCommandType.None
  
  ExecuteStream throws an InvalidOperationException.

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

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

- XmlCommandType = OracleXmlCommandType.Insert, OracleXmlCommandType.Update, or OracleXmlCommandType.Delete.
  
  The value of the CommandText property is an XML document, and ExecuteXmlReader saves the changes in that XML document to the table or view that is specified in the XmlSaveProperties property. An empty .NET XmlTextReader object is returned.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommand Class
- OracleCommand Members
- Oracle XML DB Developer’s Guide
- http://otn.oracle.com/
OracleCommandBuilder Class

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

Class Inheritance
System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      System.Data.Common.DbCommandBuilder (ADO.NET 2.0 only)
        OracleDataAccess.Client.OracleCommandBuilder

Declaration
// ADO.NET 2.0: C#
public sealed class OracleCommandBuilder : DbCommandBuilder

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()
    {
        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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommandBuilder Members
- OracleCommandBuilder Constructors
- OracleCommandBuilder Static Methods
- OracleCommandBuilder Properties
- OracleCommandBuilder Public Methods
- OracleCommandBuilder Events
OracleCommandBuilder Members

OracleCommandBuilder members are listed in the following tables.

OracleCommandBuilder Constructors
OracleCommandBuilder constructors are listed in Table 5–8.

Table 5–8 OracleCommandBuilder Constructors

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

OracleCommandBuilder Static Methods
OracleCommandBuilder static methods are listed in Table 5–9.

Table 5–9 OracleCommandBuilder Static Methods

<table>
<thead>
<tr>
<th>Method</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 System.Object (Overloaded)</td>
</tr>
</tbody>
</table>

OracleCommandBuilder Properties
OracleCommandBuilder properties are listed in Table 5–10.

Table 5–10 OracleCommandBuilder Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Inherited from System.ComponentModel.Component</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>CatalogLocation</td>
<td>Not Supported</td>
</tr>
<tr>
<td>CatalogSeparator</td>
<td>Not Supported</td>
</tr>
<tr>
<td>ConflictOption</td>
<td>Not Supported</td>
</tr>
<tr>
<td>DataAdapter</td>
<td>Indicates the OracleDataAdapter for which the SQL statements are generated</td>
</tr>
<tr>
<td>QuotePrefix</td>
<td>Specifies the beginning character or characters used to specify database objects whose names contain special characters such as spaces or reserved words Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>QuoteSuffix</td>
<td>Specifies the ending character or characters used to specify database objects whose names contain special characters such as spaces or reserved words Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>SchemaSeparator</td>
<td>Specifies the character to be used for the separator between the schema identifier and other identifiers Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
</tbody>
</table>
OracleCommandBuilder Class

OracleCommandBuilder Public Methods
OracleCommandBuilder public methods are listed in Table 5–11.

Table 5–11 OracleCommandBuilder Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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 System.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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.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>InitializeComponent</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>QuoteIdentifier</td>
<td>Returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>RefreshSchema</td>
<td>Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements</td>
</tr>
<tr>
<td>UnquoteIdentifier</td>
<td>Returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

OracleCommandBuilder Events
The OracleCommandBuilder event is listed in Table 5–12.

Table 5–12 OracleCommandBuilder Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
  - 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 1-4
- 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 1-4
- OracleCommandBuilder Class
- OracleCommandBuilder Members
OracleCommandBuilder Static Methods

OracleCommandBuilder static methods are listed in Table 5–13.

<table>
<thead>
<tr>
<th>Method</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 System.Object (Overloaded)</td>
</tr>
</tbody>
</table>

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

// 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

When DeriveParameters is used to populate the Parameter collection of an OracleCommand Object that represents a stored function, the return value of the function is bound as the first parameter (at position 0 of the OracleParameterCollection).

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 database round-trip and should only be used during design time. To avoid unnecessary database 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:

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

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

\<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:

```
\"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 application may also modify the metadata information of the parameters before execution. For example, the Size property of the OracleParameter may be modified for PL/SQL character and string types to optimize the execution of the stored procedure.

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.

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

class DeriveParametersSample
{
    static void Main()
    {
```
// Create the PL/SQL Stored Procedure MyOracleStoredProc as indicated in
// the preceding Database Setup

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);

// The PL/SQL stored procedure MyOracleStoredProc has one IN and
// one OUT parameter.  Set the Value for the IN parameter.
cmd.Parameters[0].Value = "MyText";

// The application may modify the other OracleParameter properties also
// This sample uses the default Size for the IN parameter and modifies
// the Size for the OUT parameter

// The default size for OUT VARCHAR2 is 4000
// Prints 'cmd.Parameters[1].Size  = 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();
}
Example

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

OracleCommandBuilder properties are listed in Table 5–14.

**Table 5–14 OracleCommandBuilder Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Inherited from System.ComponentModel.Component</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>CatalogLocation</td>
<td>Not Supported</td>
</tr>
<tr>
<td>CatalogSeparator</td>
<td>Not Supported</td>
</tr>
<tr>
<td>ConflictOption</td>
<td>Not Supported</td>
</tr>
<tr>
<td>DataAdapter</td>
<td>Indicates the OracleDataAdapter for which the SQL statements are generated</td>
</tr>
<tr>
<td>QuotePrefix</td>
<td>Specifies the beginning character or characters used to specify database objects whose names contain special characters such as spaces or reserved words. <strong>Supported Only in ADO.NET 2.0-Compliant ODP.NET</strong></td>
</tr>
<tr>
<td>QuoteSuffix</td>
<td>Specifies the ending character or characters used to specify database objects whose names contain special characters such as spaces or reserved words. <strong>Supported Only in ADO.NET 2.0-Compliant ODP.NET</strong></td>
</tr>
<tr>
<td>SchemaSeparator</td>
<td>Specifies the character to be used for the separator between the schema identifier and other identifiers. <strong>Supported Only in ADO.NET 2.0-Compliant ODP.NET</strong></td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
   - OracleCommandBuilder Class
   - OracleCommandBuilder Members

CatalogLocation

This property is not supported.

Declaration
// ADO.NET 2.0: C#
public override CatalogLocation CatalogLocation {get; set;}

Exceptions
NotSupportedException - This property is not supported.

Remarks
This property is not supported.

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

CatalogSeparator

This property is not supported.

Declaration
// ADO.NET 2.0: C#
public override string CatalogSeparator {get; set;}

Exceptions
NotSupportedException - This property is not supported.

Remarks
This property is not supported.

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

ConflictOption

This property is not supported.
Declaration

// ADO.NET 2.0: C#
public override string ConflictOption {get; set;}

Exceptions

NotSupportedException - This property is not supported.

Remarks

This property is not supported.

See Also:

■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleCommandBuilder Class
■ OracleCommandBuilder Members

DataAdapter

This property indicates the OracleDataAdapter object for which the SQL statements are generated.

Declaration

// C#
OracleDataAdapter DataAdapter {get; set;}

Property Value

An OracleDataAdapter object.

Remarks

Default = null

See Also:

■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleCommandBuilder Class
■ OracleCommandBuilder Members

QuotePrefix

This property specifies the beginning character or characters used to specify database objects whose names contain special characters such as spaces or reserved words.

Declaration

// ADO.NET 2.0: C#
public override string QuotePrefix {get; set;}

Property Value

The beginning character or characters to use. The default value is "\"".
Remarks
This property is independent of any OracleConnection or OracleCommand objects.

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

QuoteSuffix
This property specifies the ending character or characters used to specify database objects whose names contain special characters such as spaces or reserved words.

Declaration
// ADO.NET 2.0: C#
public override string QuoteSuffix {get; set;}

Property Value
The ending character or characters to use. The default value is "\"".

Remarks
This property is independent of any OracleConnection or OracleCommand objects.

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

SchemaSeparator
This property specifies the character to be used for the separator between the schema identifier and other identifiers.

Declaration
// ADO.NET 2.0: C#
public override string SchemaSeparator {get; set;}

Property Value
The character to be used as the schema separator.

Exceptions
NotSupportedException - The input value is not a dot (.)

Remarks
The default schema separator is a dot (.). The only acceptable value for this property is a dot (.)
This property is independent of any `OracleConnection` or `OracleCommand` objects.

**Example**

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

class SchemaSeparatorSample
{
    static void Main(string[] args)
    {
        try
        {
            OracleCommandBuilder cmdBuilder = new OracleCommandBuilder();

            //schemaSeparator is dot(.)
            Console.WriteLine("schemaSeparator is {0}",
                cmdBuilder.SchemaSeparator);

            //set the schemaseparator, only "." is allowed.
            cmdBuilder.SchemaSeparator = ".";

            // the only acceptable value for this property is a dot (.)
            // Hence the following line will throw NotSupportedException
            cmdBuilder.SchemaSeparator = "!";
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            Console.WriteLine(ex.StackTrace);
        }
    }
}
```

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleCommandBuilder Class`
- `OracleCommandBuilder Members`
OracleCommandBuilder public methods are listed in Table 5–15.

**Table 5–15  OracleCommandBuilder Public Methods**

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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 System.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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>QuoteIdentifier</td>
<td>Returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>RefreshSchema</td>
<td>Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements</td>
</tr>
<tr>
<td>UnquoteIdentifier</td>
<td>Returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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();
OracleCommandBuilder Class

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 1-4
- 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 1-4
- 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();
OracleCommandBuilder Public Methods

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 1-4
- `OracleCommandBuilder` Class
- `OracleCommandBuilder Members`

QuotedIdentifier
This method returns the correct quoted form of the provided unquoted identifier, with any embedded quotes in the identifier properly escaped.

Declaration
// ADO.NET 2.0: C#
public override string QuoteIdentifier(string unquotedIdentifier);

Parameters
- **UnquotedIdentifier**
  
  An unquoted identifier string.

Return Value
The quoted version of the identifier. Embedded quotes within the identifier are properly escaped.

Exceptions
`ArgumentNullException` - The input parameter is null.

Remarks
This method is independent of any `OracleConnection` or `OracleCommand` objects.

Example
// C#

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

class QuoteIdentifierSample
{
    static void Main(string[] args)
    {
        OracleCommandBuilder builder = new OracleCommandBuilder();
string quoteIdentifier = builder.QuoteIdentifier("US\"ER");

    //quoteIdentifier for "US\"ER" is {(\"US\"\"ER\")
    Console.WriteLine("quoteIdentifier is {0}" , quoteIdentifier);
}
}

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

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

Declaration
// ADO.NET 2.0: C#
public override void RefreshSchema();

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

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

UnquoteIdentifier
This method returns the correct unquoted form of the provided quoted identifier, removing any escape notation for quotes embedded in the identifier.

Declaration
// ADO.NET 2.0: C#
public override string UnquoteIdentifier(string quotedIdentifier);

Parameters
- quotedIdentifier
  The quoted string identifier.

Return Value
The unquoted identifier, with escape notation for any embedded quotes removed.

Exceptions
- ArgumentNullException - The input parameter is null.
- ArgumentException - The input parameter is empty.
Remarks
This method is independent of any OracleConnection or OracleCommand objects.

Example

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

class UnQuoteIdentifierSample
{
    static void Main(string[] args)
    {
        //create an OracleCommandBuilder object.
        OracleCommandBuilder builder = new OracleCommandBuilder();

        string identifier = "US""ER";
        Console.WriteLine("Identifier is {0}", identifier);

        //quote the identifier
        string quoteIdentifier = builder.QuoteIdentifier(identifier);
        Console.WriteLine("QuotedIdentifier is {0}", quoteIdentifier);

        string unquoteIdentifier = builder.UnquoteIdentifier(quoteIdentifier);
        Console.WriteLine("UnquotedIdentifier is {0}", unquoteIdentifier);
    }
}
```

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleCommandBuilder Class
- OracleCommandBuilder Members
OracleCommandBuilder Events

The `OracleCommandBuilder` event is listed in Table 5–16.

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

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

An OracleConnection object represents a connection to an Oracle database.

Class Inheritance
System.Object
    System.MarshalByRefObject
        System.ComponentModel.Component
            System.Data.Common.DbConnection (ADO.NET 2.0 only)

Declaration
// ADO.NET 2.0: C#
public sealed class OracleConnection : DbConnection, 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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Members
- OracleConnection Constructors
- OracleConnection Static Properties
- OracleConnection Static Methods
- OracleConnection Properties
- OracleConnection Public Methods
- OracleConnection Events
OracleConnection Members

OracleConnection members are listed in the following tables.

OracleConnection Constructors
OracleConnection constructors are listed in Table 5–17.

Table 5–17 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 Properties
The OracleConnection static property is listed in Table 5–19.

Table 5–18 OracleConnection Static Property

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsAvailable</td>
<td>Indicates whether or not the implicit database connection is available for use</td>
</tr>
</tbody>
</table>

OracleConnection Static Methods
The OracleConnection static methods are listed in Table 5–19.

Table 5–19 OracleConnection Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>ClearPool</td>
<td>Clears the connection pool that is associated with the provided OracleConnection object. Not supported in a .NET stored procedure</td>
</tr>
<tr>
<td>ClearAllPools</td>
<td>Clears all connections from all the connection pools Not supported in a .NET stored procedure</td>
</tr>
</tbody>
</table>

OracleConnection Properties
OracleConnection properties are listed in Table 5–20.

Table 5–20 OracleConnection Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionName</td>
<td>Specifies the action name for the connection</td>
</tr>
<tr>
<td>ClientId</td>
<td>Specifies the client identifier for the connection</td>
</tr>
<tr>
<td>ClientInfo</td>
<td>Specifies the client information for the connection</td>
</tr>
<tr>
<td>ConnectionString</td>
<td>Specifies connection information used to connect to an Oracle database</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Indicates the maximum amount of time that the Open method can take to obtain a pooled connection before the request is terminated</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Database</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>
Table 5–20 (Cont.) OracleConnection Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatabaseDomainName</td>
<td>Specifies the name of the database domain to which the connection is set</td>
</tr>
<tr>
<td>DatabaseName</td>
<td>Specifies the name of the database to which the connection is set</td>
</tr>
<tr>
<td>DataSource</td>
<td>Specifies the Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect</td>
</tr>
<tr>
<td>HostName</td>
<td>Specifies the name of the host to which the connection is set</td>
</tr>
<tr>
<td>InstanceName</td>
<td>Specifies the name of the instance to which the connection is set</td>
</tr>
<tr>
<td>ModuleName</td>
<td>Specifies the module name for the connection</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>ServiceName</td>
<td>Specifies the name of the service to which the connection is set</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>State</td>
<td>Specifies the current state of the connection</td>
</tr>
<tr>
<td>StatementCacheSize</td>
<td>Specifies the current size of the statement cache associated with this connection</td>
</tr>
</tbody>
</table>

OracleConnection Public Methods

OracleConnection public methods are listed in Table 5–21.

Table 5–21 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></td>
<td><em>Not supported in a .NET stored procedure for context connection</em></td>
</tr>
<tr>
<td>ChangeDatabase</td>
<td><em>Not Supported</em></td>
</tr>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleConnection object</td>
</tr>
<tr>
<td></td>
<td><em>Not supported in a .NET stored procedure</em></td>
</tr>
<tr>
<td>Close</td>
<td>Closes the database connection</td>
</tr>
<tr>
<td>CreateCommand</td>
<td>Creates and returns an OracleCommand object</td>
</tr>
<tr>
<td></td>
<td>Associated with the OracleConnection object</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>EnlistDistributedTransaction</td>
<td>Enables applications to explicitly enlist in a specified distributed transaction</td>
</tr>
<tr>
<td></td>
<td><em>Not supported in a .NET stored procedure</em></td>
</tr>
<tr>
<td>EnlistTransaction</td>
<td>Enables applications to enlist in a specified distributed transaction</td>
</tr>
<tr>
<td></td>
<td><em>Supported Only in ADO.NET 2.0-Compliant ODP.NET</em></td>
</tr>
<tr>
<td></td>
<td>Not supported in a .NET stored procedure</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>FlushCache</td>
<td>Flushes all updates and deletes made through REF objects retrieved using this connection</td>
</tr>
</tbody>
</table>
### OracleConnection Members

**OracleConnection Events**

OracleConnection events are listed in Table 5-22.

#### Table 5-22 OracleConnection Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Failover</td>
<td>An event that is triggered when an Oracle failover occurs</td>
</tr>
<tr>
<td></td>
<td><em>Not supported in a .NET stored procedure</em></td>
</tr>
<tr>
<td>HAEvent</td>
<td>An event that is triggered when an HA event 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 1-4
- 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 1-4
  - OracleConnection Class
  - OracleConnection Members

OracleConnection()

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

**Declaration**

// C#
public OracleConnection();

**Remarks**

The properties for OracleConnection are set to the following default values:

- **ConnectionString** = empty string
- **ConnectionTimeout** = 15 (default value of 0 is used for the implicit database connection)
- **DataSource** = empty string
- **ServerVersion** = empty string

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

OracleConnection(String)

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

**Declaration**

// C#
public OracleConnection(String connectionString);
Parameters

- `connectionString`
  
  The connection information used to connect to the Oracle database.

Remarks

The `ConnectionString` property is set to the supplied `connectionString`. The `ConnectionString` property is parsed and an exception is thrown if it contains invalid connection string attributes or attribute values.

The properties of the `OracleConnection` object default to the following values unless they are set by the connection string:

- `ConnectionString` = empty string
- `ConnectionTimeout` = 15 (default value of 0 is used for the implicit database connection)
- `DataSource` = empty string
- `ServerVersion` = empty string

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnection` Class
- `OracleConnection` Members
OracleConnection Static Properties

The OracleConnection static property is listed in Table 5–23.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsAvailable</td>
<td>Indicates whether or not the implicit database connection is available for use</td>
</tr>
</tbody>
</table>

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

IsAvailable

This property indicates whether or the implicit database connection is available for use.

Declaration

// C#
public static bool IsAvailable {get;}

Property Value

Returns true if the implicit database connection is available for use.

Remarks

The availability of the implicit database connection can be checked at run time through this static property. When Oracle Data Provider for .NET is used within a .NET stored procedure, this property always returns true. Otherwise, false is returned.

To obtain an OracleConnection object in a .NET stored procedure that represents the implicit database connection, set the ConnectionString property of the OracleConnection object to "context connection=true" and invoke the Open method.

Note that not all features that are available for an explicit user connection are available for an implicit database connection. See "Implicit Database Connection" on page 4-2 for details.

Example

// C# (Library/DLL)
using System;
using Oracle.DataAccess.Client;

public class IsAvailableSample
{
    static void MyStoredProcedure()
    {
        OracleConnection con = new OracleConnection();
        if (OracleConnection.IsAvailable)
        {
            // This function is invoked as a stored procedure...
// Obtain the implicit database connection by setting
//   "context connection=true" in the connection string
con.ConnectionString = "context connection=true";
}
else
{
    // This function is not invoked as a stored procedure
    // Set the connection string for a normal client connection
    con.ConnectionString = "user id=scott;password=tiger;data source=oracle";
}
con.Open();
Console.WriteLine("connected!");
}
OracleConnection Static Methods

The OracleConnection static methods are listed in Table 5–24.

Table 5–24  OracleConnection Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>ClearPool</td>
<td>Clears the connection pool that is associated with the provided OracleConnection object.</td>
</tr>
<tr>
<td></td>
<td>Not supported in a .NET stored procedure</td>
</tr>
<tr>
<td>ClearAllPools</td>
<td>Clears all connections from all the connection pools</td>
</tr>
<tr>
<td></td>
<td>Not supported in a .NET stored procedure</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members

ClearPool

This method clears the connection pool that is associated with the provided OracleConnection object.

Declaration

// C#
public static void ClearPool(OracleConnection connection);

Remarks

When this method is invoked, all idle connections are closed and freed from the pool. Currently used connections are not discarded until they are returned to the pool.

The ClearPool method should be invoked only when valid connections can be created (that is, the database is up and can be connected to). Otherwise, the ClearPool method may just create invalid connections to a downed database instance. Assuming valid database connections, a ClearPool invocation creates a connection pool with usable connections. Therefore, connection requests succeed even after the invocation of this method. Connections created after this method invocation are not cleared unless another invocation is made.

This method can be invoked with an OracleConnection object before opening the connection as well as after, provided the ConnectionString is properly set.

Exceptions

InvalidOperationException – Either the connection pool cannot be found or the provided connection string is invalid.

Example

// C#
// Sample demonstrating the use of ClearPool API in OracleConnection class

using System;
using Oracle.DataAccess.Client;

class ClearPoolSample
{
    static void Main()
    {
        Console.WriteLine("Running ClearPool sample...");
        // Set the connection string
        string strConn = "User Id=scott;Password=tiger;Data Source=oracle;" +
                         "Min pool size=5;";
        OracleConnection conn = new OracleConnection(strConn);

        // Open the connection
        conn.Open();

        // Clears the connection pool associated with connection 'conn'
        OracleConnection.ClearPool (conn);

        // This connection will be placed back into the pool
        conn.Close();

        // Open the connection again to create additional connections in the pool
        conn.Open();

        // Create a new connection object
        OracleConnection connNew = new OracleConnection(strConn);

        // Clears the pool associated with Connection 'connNew'
        // Since the same connection string is set for both the connections, 
        // connNew and conn, they will be part of the same connection pool.
        // We need not do an Open() on the connection object before calling
        // ClearPool
        OracleConnection.ClearPool (connNew);

        // cleanup
        conn.Close();
        Console.WriteLine("Done!");
    }
}

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

ClearAllPools

This method clears all connections from all the connection pools.

Declaration
// C#
public static void ClearAllPools();

Remarks
This call is analogous to calling ClearPool for all the connection pools that are 
created for the application.
Exceptions

InvalidOperationException – No connection pool could be found for the application.

Example

// C#
// Sample demonstrating the use of ClearAllPools API in OracleConnection class

using System;
using Oracle.DataAccess.Client;

class ClearAllPoolsSample
{
    static void Main()
    {
        Console.WriteLine("Running ClearAllPools sample...");
        // Set the connection string
        string strConn = "User Id=scott;Password=tiger;Data Source=oracle;" + "Min pool size=5;";
        OracleConnection conn = new OracleConnection(strConn);

        // Create another connection object with a different connection string
        string strConnNew = "User Id=scott;Password=tiger;Data Source=oracle;";
        OracleConnection connNew = new OracleConnection(strConnNew);

        // Open the connections. Separate pools are created for conn and connNew
        conn.Open();
        connNew.Open();

        // Clears the pools associated with conn and connNew
        OracleConnection.ClearAllPools();

        // cleanup
        conn.Close();
        connNew.Close();
        Console.WriteLine("Done!");
    }
}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
- "ClearPool" on page 5-73
OracleConnection Properties

OracleConnection properties are listed in Table 5–25

Table 5–25 OracleConnection Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionName</td>
<td>Specifies the action name for the connection</td>
</tr>
<tr>
<td>ClientId</td>
<td>Specifies the client identifier for the connection</td>
</tr>
<tr>
<td>ClientInfo</td>
<td>Specifies the client information for the connection</td>
</tr>
<tr>
<td>ConnectionString</td>
<td>Specifies connection information used to connect to an Oracle database</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Indicates the maximum amount of time that the Open method can take to obtain a pooled connection before the request is terminated</td>
</tr>
<tr>
<td>ConnectionType</td>
<td>Determines whether a particular connection object is associated with a TimesTen database connection, an Oracle database connection, or no physical connection</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Database</td>
<td>Not Supported</td>
</tr>
<tr>
<td>DatabaseDomainName</td>
<td>Specifies the name of the database domain to which the connection is set</td>
</tr>
<tr>
<td>DatabaseName</td>
<td>Specifies the name of the database to which the connection is set</td>
</tr>
<tr>
<td>DataSource</td>
<td>Specifies the Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect</td>
</tr>
<tr>
<td>HostName</td>
<td>Specifies the name of the host to which the connection is set</td>
</tr>
<tr>
<td>InstanceName</td>
<td>Specifies the name of the instance to which the connection is set</td>
</tr>
<tr>
<td>ModuleName</td>
<td>Specifies the module name for the connection</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>ServiceName</td>
<td>Specifies the name of the service to which the connection is set</td>
</tr>
<tr>
<td>Site</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>State</td>
<td>Specifies the current state of the connection</td>
</tr>
<tr>
<td>StatementCacheSize</td>
<td>Specifies the current size of the statement cache associated with this connection</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members

**ActionName**

This property specifies the action name for the connection.

**Declaration**

```csharp
// C#
```
public string ActionName {set;}

**Property Value**
The string to be used as the action name.

**Remarks**
The default value is `null`.

Using the `ActionName` property allows the application to set the action name in the application context for a given `OracleConnection` object.

The `ActionName` property is reset to `null` when the `Close` or `Dispose` method is called on the `OracleConnection` object.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnection Class`
- `OracleConnection Members`
- "Client Identifier and End-to-End Tracing" on page 3-17
- *Oracle Database Security Guide*

**ClientId**
This property specifies the client identifier for the connection.

**Declaration**
```csharp
// C#
public string ClientId {set;}
```

**Property Value**
The string to be used as the client identifier.

**Remarks**
The default value is `null`.

Using the `ClientId` property allows the application to set the client identifier in the application context for a given `OracleConnection` object.

Setting `ClientId` to `null` resets the client identifier for the connection. `ClientId` is set to `null` when the `Close` or `Dispose` method is called on the `OracleConnection` object.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnection Class`
- `OracleConnection Members`
- "Client Identifier and End-to-End Tracing" on page 3-17
- *Oracle Database Security Guide*
**ClientInfo**

This property specifies the client information for the connection.

**Declaration**

```csharp
// C#
public string ClientInfo {set;}
```

**Property Value**

The string to be used as the client information.

**Remarks**

The default value is `null`.

Using the `ClientInfo` property allows the application to set the client information in the application context for a given `OracleConnection` object.

The `ClientInfo` property is reset to `null` when the `Close` or `Dispose` method is called on the `OracleConnection` object.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnection` Class
- `OracleConnection Members`
- "Client Identifier and End-to-End Tracing" on page 3-17
- `Oracle Database Security Guide`

**ConnectionString**

This property specifies connection information used to connect to an Oracle database.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override 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:
'User Id=scott;password=tiger;data source=oracle'

If the `ConnectionString` is not in a proper format, an exception is thrown. All spaces are ignored unless they are within double quotes.

When the `ConnectionString` property is set, the `OracleConnection` object immediately parses the string for errors. An ArgumentException is thrown if the `ConnectionString` contains invalid attributes or invalid values. Attribute values for `User Id`, `Password`, `Proxy User Id`, `Proxy Password`, and `Data Source` (if provided) are not validated until the `Open` method is called.

The connection must be closed to set the `ConnectionString` property. When the `ConnectionString` property is reset, all previously set values are reinitialized to their default values before the new values are applied.

Starting with ODP.NET 11.1, password and proxy password connection string attribute values are accepted as case-sensitive strings. Thus, they are passed to the database for authentication in the case provided in the connection string. Therefore, if the database is configured to support case-sensitive passwords, passwords must be passed in the correct case.

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`.

**Remarks (.NET Stored Procedure)**

To obtain an `OracleConnection` object in a .NET stored procedure that represents the implicit database connection, set the `ConnectionString` property of the `OracleConnection` object to "context connection=true" and invoke the `Open` method. Other connection string attributes cannot be used in conjunction with "context connection" when it is set to true.

**Supported Connection String Attributes**

Table 5–26 lists the supported connection string attributes.

**Table 5–26  Supported Connection String Attributes**

<table>
<thead>
<tr>
<th>Connection String Attribute</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Lifetime</td>
<td>Maximum life time (in seconds) of the connection.</td>
<td>0</td>
</tr>
</tbody>
</table>

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 of. 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 connections in the pool below the `Min Pool Size`.  

Oracle Data Provider for .NET Classes  5-79
### Table 5–26 (Cont.) Supported Connection String Attributes

<table>
<thead>
<tr>
<th>Connection String Attribute</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Timeout</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. (The default value is 0 for the implicit database connection in a .NET stored procedure.)</td>
<td>15</td>
</tr>
<tr>
<td>Context Connection</td>
<td>Returns an implicit database connection if set to true. An implicit database connection can only be obtained from within a .NET stored procedure. Other connection string attributes cannot be used in conjunction with &quot;context connection&quot; when it is set to true. Supported in a .NET stored procedure only</td>
<td>false</td>
</tr>
<tr>
<td>Data Source</td>
<td>Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.</td>
<td>empty string</td>
</tr>
<tr>
<td>DBA Privilege</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>
<td>empty string</td>
</tr>
<tr>
<td>Decr Pool Size</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>
<td>1</td>
</tr>
<tr>
<td>Enlist</td>
<td>Controls the enlistment behavior and capabilities of a connection in context of COM+ transactions or System.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. If this attribute is set to dynamic, applications can dynamically enlist in distributed transactions. This attribute can be set to true, false, yes, no, or dynamic.</td>
<td>true</td>
</tr>
</tbody>
</table>
### Table 5–26 (Cont.) Supported Connection String Attributes

<table>
<thead>
<tr>
<th>Connection String Attribute</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA Events</td>
<td>Enables ODP.NET connection pool to proactively remove connections from the pool when an Oracle RAC service, service member, or node goes down. This feature can only be used against an Oracle RAC database and only if &quot;pooling=true&quot;. This attribute can be set to true, false, yes, or no.</td>
<td>false</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>Enables ODP.NET connection pool to balance work requests across Oracle RAC instances based on the load balancing advisory and service goal. This feature can only be used against an Oracle RAC database and only if &quot;pooling=true&quot;. This attribute can be set to true, false, yes, or no.</td>
<td>false</td>
</tr>
<tr>
<td>Incr Pool Size</td>
<td>Number of new connections to be created when all connections in the pool are in use. 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>
<td>5</td>
</tr>
<tr>
<td>Max Pool Size</td>
<td>Maximum number of connections in a pool. This attribute specifies the maximum number of connections allowed in the particular pool used by that OracleConnection. Simply changing this attribute in the connection string does not change the Max Pool Size restriction on a currently existing pool. Doing so simply creates a new pool with a different Max Pool Size restriction. This attribute must be set to a value greater than the Min Pool Size. This value is ignored unless Pooling is turned on.</td>
<td>100</td>
</tr>
<tr>
<td>Metadata Pooling</td>
<td>Caches metadata information. This attribute indicates whether or not metadata information for executed queries are cached for improved performance.</td>
<td>True</td>
</tr>
<tr>
<td>Min Pool Size</td>
<td>Minimum number of connections in a pool. This attribute specifies the minimum number of connections to be maintained by the pool during its entire lifetime. Simply changing this attribute in the connection string does not change the Min Pool Size restriction on a currently existing pool. Doing so simply creates a new pool with a different Min Pool Size restriction. This value is ignored unless Pooling is turned on.</td>
<td>1</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the user specified by User Id. This attribute specifies an Oracle user's password. Password is case-sensitive by default for Oracle Database 11g release 1 (11.1) and later.</td>
<td>empty string</td>
</tr>
</tbody>
</table>
**Table 5–26  (Cont.) Supported Connection String Attributes**

<table>
<thead>
<tr>
<th>Connection String Attribute</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persist</td>
<td>Retrieval of the 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.</td>
<td>false</td>
</tr>
<tr>
<td>Security Info</td>
<td>Connection pooling. This attribute specifies whether or not 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.</td>
<td>true</td>
</tr>
<tr>
<td>Promotable</td>
<td>Promotable to distributed transaction or not.</td>
<td>promotable</td>
</tr>
<tr>
<td>Transaction</td>
<td>If &quot;promotable&quot; is specified, the first and all subsequent connections opened in the same TransactionScope enlist in the same distributed transaction. If &quot;local&quot; is specified, the first connection opened in the TransactionScope uses a local transaction.</td>
<td></td>
</tr>
</tbody>
</table>
| Proxy User Id               | User name of the proxy user. This connection string attribute specifies the middle-tier user, or the proxy user, who establishes a connection on behalf of a client user specified by the User Id attribute. ODP.NET attempts to establish a proxy connection if either the Proxy User Id or the Proxy Password attribute is set to a non-empty string. For the proxy user to connect to an Oracle database using operating system authentication, the Proxy User Id must be set to "/".
The Proxy Password is ignored in this case. The User Id cannot be set to "/" when establishing proxy connections. The case of this attribute value is preserved. | empty string  |
| Proxy Password              | Password of the proxy user. This connection string attribute specifies the password of the middle-tier user or the proxy user. This user establishes a connection on behalf of a client user specified by the User Id attribute. ODP.NET attempts to establish a proxy connection if either the Proxy User Id or the Proxy Password attribute is set to a non-empty string. The case of this attribute value is preserved if it is surrounded by double quotes. | empty string  |
| Statement Cache Purge       | Statement cache purged when the connection goes back to the pool. If statement caching is enabled, setting this attribute to true purges the Statement Cache when the connection goes back to the pool. | false         |
ConnectionTimeout

This property indicates the maximum amount of time that the Open method can take to obtain a pooled connection before the request is terminated.

Declaration

// ADO.NET 2.0: C#
public override int ConnectionTimeout {get;}

Property Value

The maximum time allowed for a pooled connection request, in seconds.

Implements

IDbConnection
OracleConnection Properties

Remarks
This property indicates the connection timeout that has been set using the ConnectionString attribute Connection TimeOut.

This property is read-only.

Remarks (.NET Stored Procedure)
There is no connection string specified by the application and a connection on the implicit database is always available, therefore, this property is set to 0.

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

ConnectionType
This property enables an ODP.NET application to determine whether a particular connection object is associated with an Oracle database connection, a TimesTen database connection, or no physical connection at all.

Declaration
// C#
public OracleConnectionType ConnectionType {get;}

Property Value
The OracleConnectionType that this connection object is associated with.

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

Database
This property is not supported.

Declaration
// ADO.NET 2.0: C#
public override string Database {get;}

Property Value
A string.

Implements
IDbConnection.Database

Remarks
This property is not supported. It always returns an empty string.
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members

**DatabaseDomainName**
This property specifies the name of the database domain that this connection is connected to.

**Declaration**
```csharp
// C#
public string DatabaseDomainName {get;}
```

**Property Value**
The database domain that this connection is connected to.

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

**DatabaseName**
This property specifies the name of the database that this connection is connected to.

**Declaration**
```csharp
// C#
public string DatabaseName {get;}
```

**Property Value**
The database that this connection is connected to.

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

**DataSource**
This property specifies the Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.

**Declaration**
```csharp
// ADO.NET 2.0: C#
public override string DataSource {get;}
```
**OracleConnection Properties**

**Property Value**
Oracle Net Services Name, Connect Descriptor, or an easy connect naming that identifies the database to which to connect.

**Remarks (.NET Stored Procedure)**
The value of this property is always an empty string for the implicit database connection.

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

**HostName**
This property specifies the name of the host that this connection is connected to.

**Declaration**
```c#
public string HostName {get;}
```

**Property Value**
The host that this connection is connected to.

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

**InstanceName**
This property specifies the name of the instance that this connection is connected to.

**Declaration**
```c#
public string InstanceName {get;}
```

**Property Value**
The instance that this connection is connected to.

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

**ModuleName**
This property specifies the module name for the connection.
Declaration
// C#
public string ModuleName {set;}

Property Value
The string to be used as the module name.

Remarks
The default value is null.

Using the ModuleName property allows the application to set the module name in the application context for a given OracleConnection object.

The ModuleName property is reset to null when the Close or Dispose method is called on the OracleConnection object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
- "Client Identifier and End-to-End Tracing" on page 3-17
- Oracle Database Security Guide

ServerVersion
This property specifies the version number of the Oracle database to which the OracleConnection has established a connection.

Declaration
// ADO.NET 2.0: C#
public override string ServerVersion {get;}

Property Value
The version of the Oracle database.

Exceptions
InvalidOperationException - The connection is closed.

Remarks
The default is an empty string.

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

ServiceName
This property specifies the name of the service that this connection is connected to.
Declaration

// C#
public string ServiceName {get;}

Property Value
The service that this connection is connected to.

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

State
This property specifies the current state of the connection.

Declaration

// ADO.NET 2.0: C#
public override 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 1-4
- OracleConnection Class
- OracleConnection Members

StatementCacheSize
This property specifies the current size of the statement cache associated with this connection.

Declaration

// C#
public int StatementCacheSize {get;}

Property Value
An integer value indicating the size of the statement cache.
Remarks
If self tuning is not enabled, then the default value of this property depends upon the statement cache size specified in the connection string, application configuration file, or the registry. If none of these values are specified, then a default value of 0 is used.

If self tuning is enabled, then the property value is adjusted automatically. Any values specified in the connection string, application configuration file, or the registry are ignored.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
OracleConnection public methods are listed in Table 5–27.

<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></td>
<td>Not supported in a .NET stored procedure for context connection</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></td>
<td>Not supported in a .NET stored procedure</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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>EnlistDistributedTransaction</td>
<td>Enables applications to explicitly enlist in a specified distributed transaction</td>
</tr>
<tr>
<td></td>
<td>Not supported in a .NET stored procedure</td>
</tr>
<tr>
<td>EnlistTransaction</td>
<td>Enables applications to enlist in a specified distributed transaction</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td></td>
<td>Not supported in a .NET stored procedure</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>FlushCache</td>
<td>Flushes all updates and deletes made through REF objects retrieved using this connection</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetSchema</td>
<td>Returns schema information for the data source of the OracleConnection</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</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 System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.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></td>
<td>Not supported in a .NET stored procedure for context connection</td>
</tr>
<tr>
<td>PurgeStatementCache</td>
<td>Flushes the Statement Cache by closing all open cursors on the database, when statement caching is enabled</td>
</tr>
</tbody>
</table>
OracleConnection Class

Table 5-27 (Cont.) OracleConnection Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 System.Object</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members

BeginTransaction

BeginTransaction methods begin local transactions.

Overload List

- BeginTransaction()
  This method begins a local transaction.
- BeginTransaction(IsolationLevel)
  This method begins a local transaction with the specified isolation level.

BeginTransaction()

This method begins a local transaction.

Declaration

// C#
public OracleTransaction BeginTransaction();

Return Value

An OracleTransaction object representing the new transaction.

Implements

IDbConnection

Exceptions

InvalidOperationException - A transaction has already been started.

Remarks

The transaction is created with its isolation level set to its default value of IsolationLevel.ReadCommitted. All further operations related to the transaction must be performed on the returned OracleTransaction object.

Remarks (.NET Stored Procedure)

Using this method in a .NET stored procedure for context connection causes a Not Supported exception.
BeginTransaction(IsolationLevel)

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

Declaration

// C#
public OracleTransaction BeginTransaction(IsolationLevel isolationLevel);

Parameters

■ isolationLevel

The isolation level for the new transaction.

Return Value

An OracleTransaction object representing the new transaction.

Implements

IDbConnection

Exceptions

InvalidOperationException - A transaction has already been started.

ArgumentException - The isolationLevel specified is invalid.

Remarks

The following isolation levels are supported: IsolationLevel.ReadCommitted and IsolationLevel.Serializable.

Although the BeginTransaction method supports the IsolationLevel.Serializable isolation level, serializable transactions are not supported when using System.Transactions and TransactionScope.

Requesting other isolation levels causes an exception.

Remarks (.NET Stored Procedure)

Using this method in a .NET stored procedure for context connection causes a NotSupported exception.

Example

// 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 1-4
- OracleConnection Class
- OracleConnection Members

ChangeDatabase

This method is not supported.

Declaration

// ADO.NET 2.0: C#
public override void ChangeDatabase(string databaseName);

Parameters

- databaseName

The name of the database that replaces the current database name.

Implements

IDbConnection.ChangeDatabase

Exceptions

NotSupportedException - Method not supported.

Remarks

This method is not supported and throws a NotSupportedException if invoked.
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.

Remarks (.NET Stored Procedure)

This method is not supported for an implicit database connection.

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();
        if (clonedCon.ConnectionString.Equals(con.ConnectionString))
            Console.WriteLine("The connection strings are the same.");
        else
            Console.WriteLine("The connection strings are different.");

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

This method closes the connection to the database.

Declaration

// ADO.NET 2.0: C#
public override 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().

CreateCommand

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

Declaration

// C#
public OracleCommand CreateCommand();

Return Value

The OracleCommand object.

Implements

IDbConnection

Example

// C#
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 1-4
- 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.

Exceptions

InvalidOperationException - The connection is part of a local transaction or the connection is closed.

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 ContexUtil.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.

Remarks (.NET Stored Procedure)
Using this method causes a Not Supported exception.

Example

Application:

// C#

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

The Enterprise Services Component DLL must be built first before building this module.
In addition, the DLL needs to be referenced appropriately when building this application.
*/

using System;
using System.EnterpriseServices;
using DistribTxnSample;

class DistribTxnSample_App
{
    static void Main()
    {
        DistribTxnSample_Comp comp = new DistribTxnSample_Comp();
        comp.DoWork();
    }
}

Component:

// C#

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

This library must be built first before the application is built.
*/

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

namespace DistribTxnSample
{
    [Transaction(TransactionOption.RequiresNew)]
    public class DistribTxnSample_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();
        }
    }
}

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
- "EnlistTransaction" on page 5-98
- http://msdn.microsoft.com/library for detailed information about this Microsoft .NET Framework 1.1 feature

EnlistTransaction

This method enlists the connection to the specified transaction.

Supported Only in ADO.NET 2.0-Compliant ODP.NET

Declaration

// C#
public override void EnlistTransaction(Transaction transaction)

Parameters

- transaction

Exceptions
InvalidOperationException - The connection is part of a local transaction or the connection is closed.

Remarks
Invocation of this method immediately enlists the connection to a transaction that is specified by the provided transaction parameter.

If OracleConnection is still associated with a distributed transaction that has not completed from a previous EnlistTransaction method invocation, calling this method will cause an exception to be thrown.

In general, for transaction enlistments to succeed, the "enlist" connection string attribute must be set to either "true" or "dynamic" before invoking the Open method. Setting the "enlist" connection string attribute to "true" will implicitly enlist the connection when the Open method is called, if the connection is within a transaction context. Setting it to "dynamic" allows the connection to dynamically enlist in transactions when an EnlistTransaction or EnlistDistributedTransaction method is called. The "enlist" attribute should be set to "false" only if the connection will never enlist in a transaction.

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleConnection Class
- OracleConnection Members
- "System.Transactions and Promotable Transactions” on page 3-25
- "EnlistDistributedTransaction” on page 5-96

FlushCache
This method flushes all updates and deletes made through REF objects retrieved using this connection.

Declaration
```csharp
// c#
public void FlushCache();
```

Exceptions
InvalidOperationException - The specified connection is not open.

Remarks
Before flushing objects, it is required that the application has explicitly started a transaction by executing the BeginTransaction method on the OracleConnection object. This is because if the object being flushed has not already been locked by the application, an exclusive lock is obtained implicitly for the object. The lock is only released when the transaction commits or rollbacks.
OracleConnection Public Methods

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

GetSchema

GetSchema methods return schema information for the data source of the OracleConnection.

Supported Only in ADO.NET 2.0-Compliant ODP.NET

Overload List
- GetSchema()
  This method returns schema information for the data source of the OracleConnection.
- GetSchema (string collectionName)
  This method returns schema information for the data source of the OracleConnection using the specified string for the collection name.
- GetSchema (string collectionName, string[] restrictions)
  This method returns schema information for the data source of the OracleConnection using the specified string for the collection name and the specified string array for the restriction values.

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

GetSchema()

This method returns schema information for the data source of the OracleConnection.

Declaration

```csharp
// ADO.NET 2.0: C#
public override DataTable GetSchema();
```

Return Value

A DataTable object.

Exceptions

InvalidOperationException – The connection is closed.

Remarks

This method returns a DataTable object that contains a row for each metadata collection available from the database.
The method is equivalent to specifying the String value "MetaDataCollections" when using the GetSchema(String) method.

**Example**

```csharp
// C#

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

class GetSchemaSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott; Password=tiger; Data Source=oracle;";
        string ProviderName = "Oracle.DataAccess.Client";

        DbProviderFactory factory = DbProviderFactories.GetFactory(ProviderName);

        using (DbConnection conn = factory.CreateConnection())
        {
            try
            {
                conn.ConnectionString = constr;
                conn.Open();

                //Get all the schema collections and write to an XML file.
                //The XML file name is Oracle.DataAccess.Client_Schema.xml
                DataTable dtSchema = conn.GetSchema();
                dtSchema.WriteXml(ProviderName + "_Schema.xml");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }
        }
    }
}
```

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members

**GetSchema (string collectionName)**

This method returns schema information for the data source of the OracleConnection using the specified string for the collection name.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override DataTable GetSchema (string collectionName);```
Parameters

collectionName

Name of the collection for which metadata is required.

Return Value

A DataTable object.

Exceptions

ArgumentException – The requested collection is not defined.

InvalidOperationException – The connection is closed.

InvalidOperationException – The requested collection is not supported by current version of Oracle database.

InvalidOperationException – No population string is specified for requested collection.

Example

// C#

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

class GetSchemaSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott; Password=tiger; Data Source=oracle;";
        string ProviderName = "Oracle.DataAccess.Client";

        DbProviderFactory factory = DbProviderFactories.GetFactory(ProviderName);

        using (DbConnection conn = factory.CreateConnection())
        {
            try
            {
                conn.ConnectionString = constr;
                conn.Open();

                //Get MetaDataCollections and write to an XML file.
                //This is equivalent to GetSchema()
                DataTable dtMetadata =
                    conn.GetSchema(DbMetaDataCollectionNames.MetaDataCollections);
                dtMetadata.WriteXml(ProviderName + "_MetaDataCollections.xml");

                //Get Restrictions and write to an XML file.
                DataTable dtRestrictions =
                    conn.GetSchema(DbMetaDataCollectionNames.Restrictions);
                dtRestrictions.WriteXml(ProviderName + "_Restrictions.xml");

                //Get DataSourceInformation and write to an XML file.
                DataTable dtDataSrcInfo =
                    conn.GetSchema(DbMetaDataCollectionNames.DataSourceInformation);
                dtDataSrcInfo.WriteXml(ProviderName + "_DataSourceInformation.xml");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
//data types and write to an XML file.
DataTable dtDataTypes =
    conn.GetSchema(DbMetadataCollectionNames.DataTypes);
dtDataTypes.WriteXml(ProviderName + "_DataTypes.xml");

//Get ReservedWords and write to an XML file.
DataTable dtReservedWords =
    conn.GetSchema(DbMetadataCollectionNames.ReservedWords);
dtReservedWords.WriteXml(ProviderName + "_ReservedWords.xml");

//Get all the tables and write to an XML file.
DataTable dtTables = conn.GetSchema("Tables");
dtTables.WriteXml(ProviderName + "_Tables.xml");

//Get all the views and write to an XML file.
DataTable dtViews = conn.GetSchema("Views");
dtViews.WriteXml(ProviderName + "_Views.xml");

//Get all the columns and write to an XML file.
DataTable dtColumns = conn.GetSchema("Columns");
dtColumns.WriteXml(ProviderName + "_Columns.xml");
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
    Console.WriteLine(ex.StackTrace);
}
}

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

GetSchema (string collectionName, string[] restrictions)

This method returns schema information for the data source of the OracleConnection using the specified string for the collection name and the specified string array for the restriction values.

Declaration

// ADO.NET 2.0: C#
public override DataTable GetSchema (string collectionName,
    string[] restrictions);

Parameters
- collectionName
   The name of the collection of metadata being retrieved.
- restrictions
   An array of restrictions that apply to the metadata being retrieved.
Return Value
A `DataTable` object.

Exception
- `ArgumentException` – The requested collection is not defined.
- `InvalidOperationException` – One of the following conditions exist:
  - The connection is closed.
  - The requested collection is not supported by the current version of Oracle database.
  - More restrictions were provided than the requested collection supports.
  - No population string is specified for requested collection.

Remarks
This method takes the name of a metadata collection and an array of `String` values that specify the restrictions for filtering the rows in the returned `DataTable`. This returns a `DataTable` that contains only rows from the specified metadata collection that match the specified restrictions.

For example, if the `Columns` collection has three restrictions (`owner`, `tablename`, and `columnname`), to retrieve all the columns for the `EMP` table regardless of schema, the `GetSchema` method must pass in at least these values: null, `EMP`.

If no restriction value is passed in, default values are used for that restriction, which is the same as passing in null. This differs from passing in an empty string for the parameter value. In this case, the empty string ("") is considered the value for the specified parameter.

`collectionName` is not case-sensitive, but restrictions (string values) are.

Example
// C#

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

class GetSchemaSample
{
    static void Main(string[] args)
    {
        string constr = "User Id=scott; Password=tiger; Data Source=oracle;";
        string ProviderName = "Oracle.DataAccess.Client";

        DbProviderFactory factory = DbProviderFactories.GetFactory(ProviderName);

        using (DbConnection conn = factory.CreateConnection())
        {
            try
            {
                conn.ConnectionString = constr;
                conn.Open();

                //Get Restrictions
                DataTable dtRestrictions =
            }
        }
    }
}
```
conn.GetSchema(DbMetaDataCollectionNames.Restrictions);

DataView dv = dtRestrictions.DefaultView;

dv.RowFilter = "CollectionName = 'Columns'";
dv.Sort = "RestrictionNumber";

for (int i = 0; i < dv.Count; i++)
    Console.WriteLine("{0} (default) {1}",
        dtRestrictions.Rows[i]["RestrictionName"],
        dtRestrictions.Rows[i]["RestrictionDefault"]);

// Set restriction string array
string[] restrictions = new string[3];

// Get all columns from all tables owned by "SCOTT"
restrictions[0] = "SCOTT";
DataTable dtAllScottCols = conn.GetSchema("Columns", restrictions);

// clear collection
for (int i = 0; i < 3; i++)
    restrictions[i] = null;

// Get all columns from all tables named "EMP" owned by any
// owner/schema
restrictions[1] = "EMP";
DataTable dtAllEmpCols = conn.GetSchema("Columns", restrictions);

// clear collection
for (int i = 0; i < 3; i++)
    restrictions[i] = null;

// Get columns named "EMPNO" from tables named "EMP",
// owned by any owner/schema
restrictions[1] = "EMP";
restrictions[2] = "EMPNO";
DataTable dtAllScottEmpCols = conn.GetSchema("Columns", restrictions);

// clear collection
for (int i = 0; i < 3; i++)
    restrictions[i] = null;

// Get columns named "EMPNO" from all
// tables, owned by any owner/schema
restrictions[2] = "EMPNO";
DataTable dtAllEmpNoCols = conn.GetSchema("Columns", restrictions);
}

catch (Exception ex)
{
    Console.WriteLine(ex.Message);
    Console.WriteLine(ex.Source);
}
}
OracleConnection Public Methods

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

GetSessionInfo

GetSessionInfo returns or refreshes an OracleGlobalization object that represents the globalization settings of the session.

Overload List:
- GetSessionInfo()
  This method returns a new instance of the OracleGlobalization object that represents the globalization settings of the session.
- GetSessionInfo(OracleGlobalization)
  This method refreshes the provided OracleGlobalization object with the globalization settings of the session.

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

GetSessionInfo()

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

Declaration

// C#
public OracleGlobalization GetSessionInfo();

Return Value

The newly created OracleGlobalization object.

Example

// C#

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

// 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());

// Clean up
    cmd.Dispose();
    con.Dispose();
}
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 1-4
- 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.

Remarks (.NET Stored Procedure)
This method is not supported in a .NET stored procedure for context connection.

---

Note: If connection pooling is enabled, then invoking the OpenWithNewPassword method also clears the connection pool. This closes all idle connections created with the old password.

---

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
- "Password Expiration" on page 3-14
OracleConnection Class

PurgeStatementCache
This method flushes the statement cache by closing all open cursors on the database, 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.

Invocation of this method purges the cached cursors that are associated with the OracleConnection. It does not purge all the cached cursors in the database.

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 1-4
- OracleConnection Class
- OracleConnection Members
- "Statement Caching" on page 3-56
- ConnectionString on page 5-78
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 = "MM-DD-RR";
        con.SetSessionInfo(info);

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

        // Clean up
        cmd.Dispose();
        con.Dispose();
    }
}
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
OracleConnection Events

OracleConnection events are listed in Table 5–28.

### Table 5–28 OracleConnection Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Failover</td>
<td>An event that is triggered when an Oracle failover occurs</td>
</tr>
<tr>
<td></td>
<td>Not supported in a .NET stored procedure</td>
</tr>
<tr>
<td>HAEvent</td>
<td>An event that is triggered when an HA event 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 1-4
- 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.

**Remarks**

The `Failover` event is raised when a connection to an Oracle instance is unexpectedly severed. The client should create an `OracleFailoverEventHandler` delegate to listen to this event.
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
- "OracleFailoverEventArgs Properties" on page 11-7
- "OracleFailoverEventHandler Delegate" on page 11-9

HAEvent

This event is triggered when an HA event occurs.

Declaration

// C#
public static event OracleHAEventHandler HAEvent;

Event Data

The event handler receives an OracleHAEventArgs object which exposes the following properties containing information about the event.

- Source
  Indicates the source of the event.
- Status
  Indicates the status of the event.
- DatabaseName
  Indicates the database name affected by this event.
- DatabaseDomainName
  Indicates the database domain name affected by this event.
- HostName
  Indicates the host name affected by this event.
- InstanceName
  Indicates the instance name affected by this event.
- ServiceName
  Indicates the service name affected by this event.
- Time
  Indicates the time of the event.

Remarks

The HAEvent is static, which means that any HA Events that happen within the application domain can trigger this event. Note that in order to receive HA event notifications, OracleConnection objects that establish connections within the application domain must have "ha events=true" in the application. Otherwise, the application never receives any HA Events.
OracleConnection Events

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
- "OracleHAEventArgs Properties" on page 8-4
- "OracleHAEventHandler Delegate" on page 8-8

InfoMessage

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

**Declaration**

```csharp
// C#
public event OracleInfoMessageEventHandler InfoMessage;
```

**Event Data**
The event handler receives an `OracleInfoMessageEventArgs` object which exposes the following properties containing information about the event.

- **Errors**
  The collection of errors generated by the data source.
- **Message**
  The error text generated by the data source.
- **Source**
  The name of the object that generated the error.

**Remarks**
In order to respond to warnings and messages from the database, the client should create an `OracleInfoMessageEventHandler` delegate to listen to this event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnection Class
- OracleConnection Members
- "OracleInfoMessageEventArgs Properties" on page 5-244
- "OracleInfoMessageEventHandler Delegate" on page 5-247

StateChange

This event is triggered when the connection state changes.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override 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 1-4
- `OracleConnection Class`
- `OracleConnection Members`
- Microsoft ADO.NET documentation for a description of `StateChangeEventHandler"
OracleDataAdapter Class

An OracleDataAdapter object represents a data provider object that populates the DataSet and updates changes in the DataSet to the Oracle database.

Class Inheritance
System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
        System.Data.Common.DbDataAdapter (ADO.NET 2.0 only)

Declaration
// C#
public sealed class OracleDataAdapter : DbDataAdapter, IDbDataAdapter

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

Example
The following example uses the OracleDataAdapter and the dataset to update theEMP table:

// C#
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
    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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Members
- OracleDataAdapter Constructors
- OracleDataAdapter Static Methods
- OracleDataAdapter Properties
- OracleDataAdapter Public Methods
- OracleDataAdapter Events
OracleDataAdapter Members

OracleDataAdapter members are listed in the following tables.

OracleDataAdapter Constructors
OracleDataAdapter constructors are listed in Table 5–29.

Table 5–29 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
The OracleDataAdapter static method is listed in Table 5–30.

Table 5–30 OracleDataAdapter Static Method

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

OracleDataAdapter Properties
OracleDataAdapter properties are listed in Table 5–31.

Table 5–31 OracleDataAdapter Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptChangesDuringFill</td>
<td>Inherited from System.Data.Common.DataAdapter</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>ContinueUpdateOnError</td>
<td>Inherited from System.Data.Common.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>Requery</td>
<td>Determines whether or not the SelectCommand is reexecuted on the next call to Fill</td>
</tr>
<tr>
<td>ReturnProviderSpecificTypes</td>
<td>Determines if the Fill method returns ODP.NET-specific values or .NET common language specification values</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>
</tbody>
</table>
OracleDataAdapter Class

OracleDataAdapter Public Methods

OracleDataAdapter public methods are listed in Table 5–32.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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>GetFillParameters</td>
<td>Inherited from System.Data.Common.DbDataAdapter</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Update</td>
<td>Inherited from System.Data.Common.DbDataAdapter</td>
</tr>
</tbody>
</table>

OracleDataAdapter Events

OracleDataAdapter events are listed in Table 5–33.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>FillError</td>
<td>Inherited from System.Data.Common.DbDataAdapter</td>
</tr>
<tr>
<td>RowUpdated</td>
<td>This event is raised when row(s) have been updated by the Update() method</td>
</tr>
</tbody>
</table>
### Table 5–33 (Cont.) OracleDataAdapter Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowUpdating</td>
<td>This event is raised when row data are about to be updated to the database</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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter()

This constructor creates an instance of an OracleDataAdapter class with no arguments.

Declaration

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

Remarks

Initial values are set for the following OracleDataAdapter properties as indicated:

- **MissingMappingAction = MissingMappingAction.Passthrough**
- **MissingSchemaAction = MissingSchemaAction.Add**

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members

OracleDataAdapter(OracleCommand)

This constructor creates an instance of an OracleDataAdapter class with the provided OracleCommand as the SelectCommand.
Declaration

```csharp
// 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:


See Also:

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

```csharp
// 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:


See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
OracleDataAdapter Static Methods

The `OracleDataAdapter` static method is listed in Table 5-34.

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

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

OracleDataAdapter properties are listed in Table 5–35.

Table 5–35  OracleDataAdapter Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptChangesDuringFill</td>
<td>Inherited from System.Data.Common.DataAdapter</td>
</tr>
<tr>
<td>Container</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>ContinueUpdateOnError</td>
<td>Inherited from System.Data.Common.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>Requery</td>
<td>Determines whether or not the SelectCommand is reexecuted on the next call to Fill</td>
</tr>
<tr>
<td>ReturnProviderSpecificTypes</td>
<td>Determines if the Fill method returns ODP.NET-specific values or .NET common language specification values</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 System.ComponentModel.Component</td>
</tr>
<tr>
<td>TableMappings</td>
<td>Inherited from System.Data.Common.DataAdapter</td>
</tr>
<tr>
<td>UpdateBatchSize</td>
<td>Specifies a value that enables or disables batch processing support, and specifies the number of SQL statements that can be executed in a single round-trip to the database Supported Only in ADO.NET 2.0-Compliant ODP.NET</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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
DeleteCommand

This property is a SQL statement or stored procedure to delete rows from an Oracle database.

**Declaration**

```csharp
// 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 1-4
- 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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
**Requery**

This property determines whether or not the `SelectCommand` is reexecuted on the next call to `Fill`.

**Declaration**

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

**Property Value**

Returns `true` if the `SelectCommand` is reexecuted on the next call to `Fill`; otherwise, returns `false`.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-130

**ReturnProviderSpecificTypes**

This property determines if the `Fill` method returns ODP.NET-specific values or .NET common language specification compliant values.

*Supported Only in ADO.NET 2.0-Compliant ODP.NET*

**Declaration**

// C#
public Boolean ReturnProviderSpecificTypes {get; set;}

**Property Value**

A value that indicates whether or not the `Fill` method returns ODP.NET-specific values.

A value of `false` indicates that the `Fill` method returns .NET common language specification compliant values.

The default is `false`.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members

**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;}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
**Property Value**
A hash table.

**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**
See the example in "OracleDataAdapter Safe Type Mapping" on page 3-127.

[See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Safe Type Mapping" on page 3-127

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

**Declaration**
```
// C#
public OracleCommand SelectCommand {get; set;}
```

**Property Value**
An OracleCommand used during the Fill call to populate the selected rows to the DataSet.

**Remarks**
Default = null

If the SelectCommand does not return any rows, no tables are added to the dataset and no exception is raised.

If the SELECT statement selects from a VIEW, no key information is retrieved when a FillSchema() or a Fill() with MissingSchemaAction.AddWithKey is invoked.

[See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-130

**UpdateBatchSize**

This property specifies a value that enables or disables batch processing support, and specifies the number of SQL statements that can be executed in a single round-trip to the database.

*Supported Only in ADO.NET 2.0-Compliant ODP.NET*

**Declaration**

// C#
public virtual int UpdateBatchSize {get; set;}

**Property Value**

An integer that returns the batch size.

**Exceptions**

ArgumentOutOfRangeException - The value is set to a number < 0.

**Remarks**

Update batches executed with large amounts of data may encounter an "PLS-00123: Program too large" error. To avoid this error, reduce the size of UpdateBatchSize to a smaller value.

For each row in the DataSet that has been modified, added, or deleted, one SQL statement will be executed on the database.

Values are as follows:

- **Value = 0**
  The data adapter executes all the SQL statements in a single database round-trip

- **Value = 1** - Default value
  This value disables batch updating and SQL statements are executed one at a time.

- **Value = n where n > 1**
  The data adapter updates n rows of data per database round-trip.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "Batch Processing Support" on page 3-29

**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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-130
OracleDataAdapter Public Methods

OracleDataAdapter public methods are listed in Table 5–36.

Table 5–36  OracleDataAdapter Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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>GetFillParameters</td>
<td>Inherited from System.Data.Common.DbDataAdapter</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Update</td>
<td>Inherited from System.Data.Common.DbDataAdapter</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
- 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**
- ArgumentException - 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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-130

**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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-130

**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**

```csharp
// 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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-130

### 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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleDataAdapter Requery Property" on page 3-130
OracleDataAdapter Events

OracleDataAdapter events are listed in Table 5–37.

Table 5–37  OracleDataAdapter Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposed</td>
<td>Inherited from System.ComponentModel.Component</td>
</tr>
<tr>
<td>FillError</td>
<td>Inherited from System.Data.Common.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 1-4
- 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
using System;
using System.Data;
using Oracle.DataAccess.Client;

class RowUpdatedSample
{
    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);
    }

    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"];null

        // 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 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleRowUpdatedEventHandler Delegate" on page 5-332

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 5-137.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataAdapter Class
- OracleDataAdapter Members
- "OracleRowUpdatingEventHandler Delegate" on page 5-340
OracleDatabase Class

An OracleDatabase object represents an Oracle Database instance.

Class Inheritance
System.Object


Declaration
// C#
public sealed class OracleDatabase : IDisposable

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;

namespace Startup
{
    class Test
    {
        static void Main()
        {
            OracleConnection con = null;
            OracleDatabase db = null;
            string constring = "dba privilege=sysdba;user id=scott;password=tiger;data source=oracle";

            try
            {
                // Open a connection to see if the DB is up
                con = new OracleConnection(constring);
                con.Open();

                Console.WriteLine("The Oracle database is already up.");
            }
            catch (OracleException ex)
            {
                // If the database is down, start up the DB
                if (ex.Number == 1034)
                {
                    Console.WriteLine("The Oracle database is down.");

                    // Create an instance of an OracleDatabase object
                    db = new OracleDatabase(constring);

                    // Start up the database
                    db.Startup();
                }
                Console.WriteLine("The Oracle database is now up.");
            }
        }
    }
}
// Executing Startup() is the same as the following:
// db.Startup(OracleDBStartupMode.NoRestriction, null, true);
// which is also the same as:
// db.Startup(OracleDBStartupMode.NoRestriction, null, false);
// db.ExecuteNonQuery(“ALTER DATABASE MOUNT”);
// db.ExecuteNonQuery(“ALTER DATABASE OPEN”);

// Dispose the OracleDatabase object
db.Dispose();
}

else
{
   Console.WriteLine(“Error: “ + ex.Message);
}
} finally
{
   // Dispose the OracleConnection object
   con.Dispose();
}
}

Requirements

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Members
- OracleDatabase Constructor
- OracleDatabase Properties
- OracleDatabase Public Methods
OracleDatabase Members

OracleDatabase members are listed in the following tables.

**OracleDatabase Constructors**

The OracleDatabase constructor is listed in Table 5–38.

*Table 5–38 OracleDatabase Constructors*

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleDatabase Constructor</td>
<td>Instantiates a new instance of OracleDatabase class using the supplied connection string</td>
</tr>
</tbody>
</table>

**OracleDatabase Properties**

The OracleDatabase properties are listed in Table 5–39.

*Table 5–39 OracleDatabase Properties*

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerVersion</td>
<td>Specifies the database version number of the Oracle Database instance to which the connection is made</td>
</tr>
</tbody>
</table>

**OracleDatabase Public Methods**

The OracleDatabase public methods are listed in Table 5–40.

*Table 5–40 OracleDatabase Public Methods*

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecuteNonQuery</td>
<td>Executes the supplied non-SELECT statement against the database</td>
</tr>
<tr>
<td>Shutdown</td>
<td>Shuts down the database (Overloaded)</td>
</tr>
<tr>
<td>Startup</td>
<td>Starts up the database (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
OracleDatabase Constructor

The OracleDatabase constructor instantiates a new instance of the OracleDatabase class using the supplied connection string.

Declaration

```csharp
// C#
public OracleDatabase(string connectionString);
```

Parameters

- `connectionString`
  The connection information used to connect to the Oracle Database instance.

Remarks

The `connectionString` follows the same format used by the OracleConnection object. However, the OracleDatabase constructor accepts only the user id, password, data source, and dba privilege connection string attributes. All other attribute values are ignored. The supplied `connectionString` must contain the dba privilege connection string attribute that is set to either SYSDBA or SYSOPER.

The OracleDatabase object creates a connection upon construction and remains connected throughout its lifetime. The connection is destroyed when the OracleDatabase object is disposed. This connection is not pooled to be used by another OracleDatabase object.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
- OracleDatabase Members
OracleDatabase Properties

The OracleDatabase properties are listed in Table 5–41.

Table 5–41 OracleDatabase Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerVersion</td>
<td>Specifies the database version number of the Oracle Database instance to which the connection is made</td>
</tr>
</tbody>
</table>

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

ServerVersion

This property returns the database version number of the Oracle Database instance to which the connection is made.

Declaration

Public string ServerVersion {get;}

Property value

Returns the database version of the Oracle Database instance.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
- OracleDatabase Members
OracleDatabase Public Methods

The OracleDatabase public methods are listed in Table 5–42.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ExecuteNonQuery</code></td>
<td>Executes the supplied non-SELECT statement against the database</td>
</tr>
<tr>
<td><code>Shutdown</code></td>
<td>Shuts down the database (Overloaded)</td>
</tr>
<tr>
<td><code>Startup</code></td>
<td>Starts up the database (Overloaded)</td>
</tr>
</tbody>
</table>

### See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
- OracleDatabase Members

#### ExecuteNonQuery

This method executes the supplied non-SELECT statement against the database.

**Declaration**

```csharp
// C#
public void ExecuteNonQuery(string sql);
```

**Exceptions**

OracleException - The command execution has failed.

**Remarks**

This method is meant for execution of DDL statements such as `ALTER DATABASE` statements to `OPEN` and `MOUNT` the database, for example. This method should not be used to execute `SQL SELECT` statements. This method does not support any parameter binding.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
- OracleDatabase Members

#### Shutdown

Shutdown methods shut down a database instance.

**Overload List**

- **Shutdown()**
  
  This method shuts down the database.

- **Shutdown(OracleDBShutdownMode, bool)**
  
  This method shuts down the database using the specified mode.
Shutdown()

This method shuts down the database.

Declaration

// C#
public void Shutdown();

Exceptions

OracleException - The database shutdown request has failed.

Remarks

This method shuts down a database instance in the OracleDBShutdownMode.Default mode. New connections are refused, and the method waits for the existing connections to end.

---

**Note:** As the shutdown is effected using the OracleDBShutdownMode.Default mode, the shutdown request may remain pending if there are open connections other than the connection created by the OracleDatabase object.

---

After the connections have closed, the method closes the database, dismounts the database, and shuts down the instance using the OracleDBShutdownMode.Final mode.

This method does not throw exceptions for cases where the database has been already closed, dismounted, or shutdown appropriately. If other errors are encountered, then an exception is thrown.

Invoking this method against an Oracle Real Application Clusters (Oracle RAC) database shuts down only that database instance to which the OracleDatabase object is connected.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
- OracleDatabase Members
- "OracleDBShutdownMode Enumeration" on page 5-358

Shutdown(OracleDBShutdownMode, bool)

This method shuts down the database instance using the specified mode.

Declaration

// C#
public void Shutdown(OracleDBShutdownMode shutdownMode, bool bCloseDismountAndFinalize);

**Parameters**

- **shutdownMode**
  A `OracleDBShutdownMode` enumeration value.

- **bCloseDismountAndFinalize**
  A boolean signifying whether the database is to be closed, dismounted, and finalized.

**Exceptions**

`OracleException` - The database shutdown request has failed.

**Remarks**

This method shuts down a database instance in the specified mode. If the `bCloseDismountAndFinalize` parameter is `true`, then the method also closes the database, dismounts the database, and shuts down the instance using the `OracleDBShutdownMode.Final` mode.

If the `bCloseDismountAndFinalize` parameter is `true`, then this method does not throw exceptions for cases where the database has been already closed, dismounted, or shutdown appropriately. If other errors are encountered, then an exception is thrown.

If the `bCloseDismountAndFinalize` parameter is `false`, then the application needs to explicitly close and dismount the database. The application can then reinvoke the method using the `OracleDBShutdownMode.Final` mode to properly shut down the database. For example, if `db` is an instance of the `OracleDatabase` class, then the application invokes the following:

1. `db.Shutdown(OracleDBShutdownMode.Default, false);`
2. `db.ExecuteNonQuery("ALTER DATABASE CLOSE NORMAL");`
3. `db.ExecuteNonQuery("ALTER DATABASE DISMOUNT");`
4. `db.Shutdown(OracleDBShutdownMode.Final);`

**Note:**

- The `OracleDBShutdownMode.Final` enumeration value should not be used as the `shutdownMode` for the initial method invocation. The `OracleDBShutdownMode.Final` mode should be used only if the database is already closed and dismounted. Otherwise, the method might wait indefinitely.

- If the specified `shutdownMode` is `OracleDBShutdownMode.Final`, then the value of the `bCloseDismountAndFinalize` input parameter is ignored, as the database should have been closed and dismounted already.

If the specified `shutdownMode` is `OracleDBShutdownMode.Abort`, then the value of the `bCloseDismountAndFinalize` input parameter is ignored, as the `Abort` mode requires the database to be closed, dismounted, and finalized.
Invoking this method against an Oracle Real Application Clusters (Oracle RAC) database shuts down only that database instance to which the OracleDatabase object is connected.

**Example**

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

namespace Shutdown
{
    class Test
    {
        static void Main()
        {
            OracleConnection con = null;
            OracleDatabase db = null;
            string constring = "user id=scott;password=tiger;data source=oracle;" + "pooling=false;dba privilege=sysdba";

            try
            {
                // Open a connection to see if the DB is up;
                con = new OracleConnection(constring);
                con.Open();

                Console.WriteLine("The Oracle database is currently up.");

                // If open succeeds, we know that the database is up.
                // We have to dispose the connection so that we can
                //   shutdown the database.
                con.Dispose();

                // Shutdown the database
                db = new OracleDatabase(constring);
                db.Shutdown();

                Console.WriteLine("The Oracle database is shut down.");

                // Executing Shutdown() above is the same as the following:
                // db.Shutdown(OracleDBShutdownMode.Default, false);
                // db.ExecuteNonQuery("ALTER DATABASE CLOSE NORMAL");
                // db.ExecuteNonQuery("ALTER DATABASE DISMOUNT");
                // db.Shutdown(OracleDBShutdownMode.Final);

                // Dispose the OracleDatabase object
                db.Dispose();
            }
            catch (OracleException ex)
            {
                Console.WriteLine("An error has occurred: {0}", ex.Message);
            }
        }
    }
}
```
Startup

Startup methods enable a user with database administrator privileges to start a database instance.

Overload List

- **Startup()**
  
  This method starts a database instance using the server-side parameter file.

- **Startup(OracleDBStartupMode, string, bool)**
  
  This method starts a database instance using the client-side parameter file.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
- OracleDatabase Members
- "OracleDBShutdownMode Enumeration" on page 5-358

**Startup()**

This method starts up the database.

**Declaration**

```csharp
// C#
public void Startup();
```

**Exceptions**

OracleException - The database startup request has failed.

**Remarks**

This method starts a database instance in the OracleDbStartupMode.Normal mode using the server-side parameter file (spfile). After the database is successfully started, this method also executes the ALTER DATABASE MOUNT and ALTER DATABASE OPEN statements.

This method does not throw exceptions for cases where the database is already mounted, opened, or started appropriately. If other errors are encountered, then an exception is thrown.
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
- OracleDatabase Members
- "OracleDBStartupMode Enumeration" on page 5-359

**Startup(OracleDBStartupMode, string, bool)**

This method starts up the database using the specified startup mode.

**Declaration**

// C#
public void Startup(OracleDbStartupMode startupMode, string pfile, bool bMountAndOpen);

**Parameters**

- **startupMode**
  An OracleDBStartupMode enumeration value.

- **pfile**
  The location and name of the client-side parameter file. For example, "c:\\admin\\init.ora".
  The name of the parameter file varies depending on the operating system. For example, it can be in mixed case or lowercase, or it can have a logical name or a variation of the name init.ora. The default location is usually ORACLE_HOME\dbs or ORACLE_HOME\database.

- **bMountAndOpen**
  A true/false value signifying whether the database is to be mounted and opened.

**Exceptions**

OracleException - The database startup request has failed.

**Remarks**

This method starts a database instance in the specified mode using the specified client-side parameter file. After the database is successfully started, and if bMountAndOpen input parameter is true, this method also executes the ALTER DATABASE MOUNT and ALTER DATABASE OPEN statements.

If bMountAndOpen is true, then this method does not throw an exception for cases where the database is already mounted, opened, or started appropriately. If other errors are encountered, then an exception is thrown.

If bMountAndOpen is false, then the database must be mounted and opened explicitly by the application. For example, if db is an instance of the OracleDatabase class, then the application invokes the following:

1. db.Startup(OracleDBStartupMode.NoRestriction, null, false);
2. db.ExecuteNonQuery("ALTER DATABASE MOUNT");
3. db.ExecuteNonQuery("ALTER DATABASE OPEN");
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDatabase Class
- OracleDatabase Members
- "OracleDBStartupMode Enumeration" on page 5-359
OracleDataReader Class

An OracleDataReader object represents a forward-only, read-only, in-memory result set.

Unlike the DataSet, the OracleDataReader object stays connected and fetches one row at a time.

The following section contain related information:

- "Obtaining LONG and LONG RAW Data" on page 3-65.
- "Obtaining Data from an OracleDataReader Object" on page 3-62.

Class Inheritance

System.Object
  System.MarshalByRefObject
    System.Data.Common.DataReader
      System.Data.Common.DbDataReader (ADO.NET 2.0 only)
        Oracle.DataAccess.Client.OracleDataReader

Declaration

// ADO.NET 2.0: C#
public sealed class OracleDataReader : DbDataReader, 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.

To minimize the number of open database cursors, OracleDataReader objects should be explicitly disposed.

Example

The following OracleDataReader example retrieves the data from the EMP table:

/* Database Setup, if you have not done so yet.
   connect scott/tiger@oracle
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,byteCodes) values

(1,'KING','SOFTWARE ENGR', '5657');
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(2,'SCOTT','MANAGER', '5960');
commit;
*/

// C#
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.WriteLine("Byte codes:");
    for (int i = 0; i < len; i++)
        Console.Write(byteCodes[i].ToString("x"));
    Console.WriteLine();
}

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

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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
The OracleDataReader static method is listed in Table 5–43.

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

OracleDataReader Properties
OracleDataReader properties are listed in Table 5–44.

<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>HiddenFieldCount</td>
<td>Gets the number of fields in the OracleDataReader that are hidden</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>IsClosed</td>
<td>Indicates whether or not 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>
<tr>
<td>RowSize</td>
<td>Gets the amount of memory the internal cache of the OracleDataReader needs to store one row of data.</td>
</tr>
<tr>
<td>VisibleFieldCount</td>
<td>Gets the number of fields in the OracleDataReader that are not hidden</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
</tbody>
</table>

OracleDataReader Public Methods
OracleDataReader public methods are listed in Table 5–45.

<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 System.MarshalByRefObject</td>
</tr>
</tbody>
</table>
### Table 5–45 (Cont.) OracleDataColumn Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Releases any resources or memory allocated by the object</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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>GetEnumerator</td>
<td>Returns an IEnumerator that can be used to iterate through the collection</td>
</tr>
<tr>
<td>GetFieldType</td>
<td>Returns the Type of the specified column</td>
</tr>
<tr>
<td>GetFirstName</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>Public Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</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>GetOracleRef</td>
<td>Returns an OracleRef object of the specified REF column</td>
</tr>
<tr>
<td>GetOracleString</td>
<td>Returns an OracleString structure of the specified column</td>
</tr>
<tr>
<td>GetOracleTimeStamp</td>
<td>Returns an OracleTimeStamp structure of the specified 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>GetProviderSpecificFieldType</td>
<td>Returns the provider-specific type of the specified column</td>
</tr>
<tr>
<td></td>
<td><em>Supported Only in ADO.NET 2.0-Compliant</em> ODP.NET</td>
</tr>
<tr>
<td>GetProviderSpecificValue</td>
<td>Returns an object that represents the underlying provider-specific value of the specified ordinal</td>
</tr>
<tr>
<td></td>
<td><em>Supported Only in ADO.NET 2.0-Compliant</em> ODP.NET</td>
</tr>
<tr>
<td>GetProviderSpecificValues</td>
<td>Returns an array of objects that represent the underlying provider-specific values</td>
</tr>
<tr>
<td></td>
<td><em>Supported Only in ADO.NET 2.0-Compliant</em> ODP.NET</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 System.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 value of an XMLType column as an instance of an .NET XmlTextReader</td>
</tr>
<tr>
<td>IsDBNull</td>
<td>Indicates whether or not the column value is null</td>
</tr>
</tbody>
</table>
### Table 5-45  (Cont.) OracleDataReader Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 System.Object</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
OracleDataReader Static Methods

The `OracleDataReader` static method is listed in Table 5–46.

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleDataReader` Class
- `OracleDataReader Members`
OracleDataReader Properties

OracleDataReader properties are listed in Table 5–47.

<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>HiddenFieldCount</td>
<td>Gets the number of fields in the OracleDataReader that are hidden</td>
</tr>
<tr>
<td></td>
<td><em>Supported Only in ADO.NET 2.0-Compliant ODP.NET</em></td>
</tr>
<tr>
<td>IsClosed</td>
<td>Indicates whether or not 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 <strong>LONG</strong> and <strong>LONG RAW</strong> 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>
<tr>
<td>RowSize</td>
<td>Gets the amount of memory the internal cache of the OracleDataReader needs to store one row of data</td>
</tr>
<tr>
<td>VisibleFieldCount</td>
<td>Gets the number of fields in the OracleDataReader that are not hidden</td>
</tr>
<tr>
<td></td>
<td><em>Supported Only in ADO.NET 2.0-Compliant ODP.NET</em></td>
</tr>
</tbody>
</table>

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

Depth

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

Declaration

```csharp
// ADO.NET 2.0: C#
public override int Depth {get;}
```

Property Value

The depth of nesting for the current row.

Implements

IDataReader
Exceptions
InvalidOperationException - The reader is closed.

Remarks
Default = 0
This property always returns zero because Oracle does not support nesting.

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

FetchSize
This property specifies the size of OracleDataReader's internal cache.

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

Property Value
A long that specifies the amount of memory (in bytes) that the OracleDataReader uses for its internal cache.

Exceptions
ArgumentException - The FetchSize value specified is invalid.

Remarks
Default = The OracleCommand's FetchSize property value.

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

The RowSize and FetchSize properties handle UDT and XMLType data differently than other scalar data types. Because only a reference to the UDT and XMLType data is stored in the ODP.NET's internal cache, the RowSize property accounts for only the memory needed for the reference (which is very small) and not the actual size of the UDT and XMLType data. Thus, applications can inadvertently fetch a large number of UDT or XMLType instances from the database in a single database round-trip. This is because the actual size of UDT and XMLType data does not count against the FetchSize, and it would require numerous UDT and XMLType references to fill up the default cache size of 131072 bytes. Therefore, when fetching UDT or XMLType data, the FetchSize property must be appropriately configured to control the number of UDT and XMLType instances that are to be fetched, rather than the amount of the actual UDT and XMLType data to be fetched.

NOTE: For LOB and LONG data types, only the sizes specified in the InitialLOBFetchSize and InitialLONGFetchSize properties are accounted for by the RowSize property in addition to the metadata and reference information that is maintained by the cache for each LOB in the select list.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
- OracleDataReader "RowSize" on page 168
- OracleCommand "ExecuteReader()" on page 5-34
- OracleCommand "RowSize" on page 5-23

FieldCount

This property returns the number of columns in the result set.

Declaration

// ADO.NET 2.0: C#
public override 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 1-4
- OracleDataReader Class
- OracleDataReader Members

HasRows

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

Declaration

// ADO.NET 2.0: C#
public override bool HasRows {get;}

Return Value

bool

Remarks

HasRows indicates whether or not 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 determine the emptiness of the `OracleDataReader` when `HasRows` property is accessed for the first time after the creation of the `OracleDataReader` object.

**Example**

```
// 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 1-4
- `OracleDataReader` Class
- `OracleDataReader` Members
- [http://msdn.microsoft.com/library](http://msdn.microsoft.com/library) for detailed information about this Microsoft .NET Framework 1.1 feature

### HiddenFieldCount

This property gets the number of fields in the `OracleDataReader` that are hidden.

**Supported Only in ADO.NET 2.0-Compliant ODP.NET**

**Declaration**

```
// C#
public int HiddenFieldCount { get; }
```
**Property Value**
The number of fields in the `OracleDataReader` that are hidden.

**Exceptions**
`InvalidOperationException` - The reader is closed.

**Remarks**
`OracleDataReader.FieldCount` and `OracleDataReader.VisibleFieldCount` return the visible field count.

See Also:
- "`Oracle.DataAccess.Client Namespace`" on page 1-4
- `OracleDataReader Class`
- `OracleDataReader Members`
- "`VisibleFieldCount`" on page 5-168
- "`FieldCount`" on page 5-162

**IsClosed**
This property indicates whether or not the data reader is closed.

**Declaration**
```
// ADO.NET 2.0: C#
public override 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 1-4
- `OracleDataReader Class`
- `OracleDataReader Members`

**Item**
This property gets the value of the column in .NET data type.

**Overload List:**
- `Item [index]`
  This property gets the .NET `Value` of the column specified by the column index.
• **Item [string]**

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

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

**Item [index]**

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

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override 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 1-4
- OracleDataReader Class
- OracleDataReader Members

**Item [string]**

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

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override object this[string columnName] {get;}
```

**Parameters**

- **columnName**
  
  The name of the column.

**Property Value**

The .NET value of the specified column.
OracleDataReader Properties

Implements
IDataRecord

Remarks
Default = Not Applicable
A case-sensitive search is made to locate the specified column by its name. If this fails, then a case-insensitive search is made.
In C#, this property is the indexer for this class.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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
For Oracle Database 10g release 2 (10.2) and later, the maximum value supported for InitialLOBFetchSize is 2 GB.
For releases prior to Oracle Database 10g release 2 (10.2), the maximum value supported for InitialLOBFetchSize is 32K.
Default is the OracleCommand.InitialLOBFetchSize, from which this value is inherited.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
- "InitialLOBFetchSize" on page 5-18 for further information on OracleCommand.InitialLOBFetchSize
- "Obtaining LOB Data" on page 3-66

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 1-4
- OracleDataReader Class
- OracleDataReader Members
- "InitialLONGFetchSize" on page 5-20 for further information on OracleCommand.InitialLONGFetchSize
- "Obtaining LONG and LONG RAW Data" on page 3-65

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.
OracleDataReader Properties

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

RowSize

This property gets the amount of memory the internal cache of the OracleDataReader needs 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

The RowSize property is set to a nonzero value when the OracleDataReader object is created. This property can be used at design time or dynamically during run time, to set the FetchSize property, based on the number of rows. For example, to enable the OracleDataReader object to fetch \( N \) rows for each database round-trip, the OracleDataReader FetchSize property can be set dynamically to \( \text{RowSize} \times N \). Note that for the FetchSize property to take effect appropriately, it must be set before the first invocation of OracleDataReader.Read() for the particular result set.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
- "FetchSize" on page 161

VisibleFieldCount

This property gets the number of fields in the OracleDataReader that are not hidden.

Supported Only in ADO.NET 2.0-Compliant ODP.NET

Declaration

```csharp
// C#
public override int VisibleFieldcount { get; }
```

Property Value

The number of fields that are not hidden.

Exceptions

InvalidOperationException - The reader is closed.
Remarks
If an application sets the AddRowid property on an OracleCommand object to true, then the application can access the RowId but it is not a visible field. If RowId is added in the select statement list, then it is a visible field. OracleDataReader.VisibleFieldCount and OracleDataReader.FieldCount always have the same value.

Example

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

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

        using (DbConnection conn = factory.CreateConnection())
        {
            conn.ConnectionString = constr;
            try
            {
                conn.Open();
                OracleCommand cmd = (OracleCommand)factory.CreateCommand();
                cmd.Connection = (OracleConnection)conn;

                //to gain access to ROWIDs of the table
                cmd.AddRowid = true;
                cmd.CommandText = "select empno, ename from emp;";
                OracleDataReader reader = cmd.ExecuteReader();

                int visFC = reader.VisibleFieldCount; //Results in 2
                int hidFC = reader.HiddenFieldCount; // Results in 1

                Console.WriteLine("Visible field count: " + visFC);
                Console.WriteLine("Hidden field count: " + hidFC);
                reader.Dispose();
                cmd.Dispose();
            } catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }
        }
    }
}
```
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
- "VisibleFieldCount" on page 5-168
- "FieldCount" on page 5-162
OracleDataReader Public Methods

OracleDataReader public methods are listed in Table 5–48.

**Table 5–48 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 System.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 System.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,</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Returns an IEnumerator that can be used to iterate through the collection</td>
</tr>
<tr>
<td>GetFieldType</td>
<td>Returns the Type of the specified column</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>Public Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</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>GetOracleRef</td>
<td>Returns an OracleRef object of the specified REF column</td>
</tr>
<tr>
<td>GetOracleString</td>
<td>Returns an OracleString structure of the specified column</td>
</tr>
<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>GetProviderSpecificFieldType</td>
<td>Returns the provider-specific type of the specified column</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>GetProviderSpecificValue</td>
<td>Returns an object that represents the underlying provider-specific value of the specified ordinal</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>GetProviderSpecificValues</td>
<td>Returns an array of objects that represent the underlying provider-specific values</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</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 System.Object class</td>
</tr>
<tr>
<td>GetValue</td>
<td>Returns the column value as a .NET type</td>
</tr>
</tbody>
</table>
Table 5–48  (Cont.) OracleDataReader Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 or not 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 System.Object</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleDataReader Class
- OracleDataReader Members

Close

This method closes the OracleDataReader.

Declaration

// ADO.NET 2.0: C#
public override 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 5-152.

See Also:

- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleDataReader Class
- OracleDataReader Members

Dispose

This method releases any resources or memory allocated by the object.

Declaration

// C#
public void Dispose();
**OracleDataReader Public Methods**

**Implements**
IDisposable

**Remarks**
The Dispose method also closes the OracleDataReader.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

**GetBoolean**

This method is not supported.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override bool GetBoolean(int index);
```

**Parameters**

- `index`
  
  The zero-based column index.

**Implements**
IDataRecord

**Exceptions**

NotSupportedException - This property is not supported.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

**GetByte**

This method returns the byte value of the specified column.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override 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
IsNull should be called to check for NULL values before calling this method.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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
// ADO.NET 2.0: C#
public override long GetBytes(int index, long fieldOffset, byte[] buffer,
    int bufferOffset, int length);

Parameters
- index
  The zero-based column index.
- fieldOffset
  The offset within the column from which reading begins (in bytes).
- buffer
  The byte array that the data is read into.
- bufferOffset
  The offset within the buffer to begin reading data into (in bytes).
- length
  The maximum number of bytes to read (in bytes).

Return Value
The number of bytes read.

Implements
IDataRecord
Exceptions

InvalidOperationException - The connection is closed, the reader is closed, 
Read() has not been called, or all rows have been read.

IndexOutOfRangeException - The column index is invalid.

InvalidCastException - The accessor method is invalid for this column type or 
the column value is NULL.

Remarks

This method returns the number of bytes read into the buffer. This may be less than 
the actual length of the field if the method has been called previously for the same 
column.

If a null reference is passed for buffer, the length of the field in bytes is returned. 
IsDBNull should be called to check for NULL values before calling this method.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

GetChar

This method is not supported.

Declaration

// ADO.NET 2.0: C#
public override long GetChar(int index);

Parameters

- index
  
The zero based column index.

Implements

IDataRecord

Exceptions

NotSupportedException - This property is not supported.

See Also:

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

// ADO.NET 2.0: C#
public override 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 1-4
- OracleDataReader Class
- OracleDataReader Members

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

**Declaration**

// ADO.NET 2.0: C#
public override 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 1-4
- OracleDataReader Class
- OracleDataReader Members

### GetDateTime

This method returns the DateTime value of the specified column.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override 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.
GetDecimal

This method returns the \texttt{decimal} value of the specified \texttt{NUMBER} column.

\textbf{Declaration}

```csharp
// ADO.NET 2.0: C#
public override decimal GetDecimal(int index);
```

\textbf{Parameters}

- \texttt{index}

  The zero-based column index.

\textbf{Return Value}

The \texttt{decimal} value of the column.

\textbf{Implements}

\texttt{IDataRecord}

\textbf{Exceptions}

- \texttt{InvalidOperationException} - The connection is closed, the reader is closed, \texttt{Read()} has not been called, or all rows have been read.
- \texttt{IndexOutOfRangeException} - The column index is invalid.
- \texttt{InvalidCastException} - The accessor method is invalid for this column type or the column value is \texttt{NULL}.

\textbf{Remarks}

\texttt{IsDBNull} should be called to check for \texttt{NULL} values before calling this method.

\textbf{See Also:}

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

GetDouble

This method returns the \texttt{double} value of the specified \texttt{NUMBER} column or \texttt{BINARY\_DOUBLE} column.

\textbf{Declaration}

```csharp
// ADO.NET 2.0: C#
public override double GetDouble(int index);
```

\textbf{Parameters}

- \texttt{index}

\textbf{See Also:}

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
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.
InvalidOperationException - 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 1-4
- OracleDataReader Class
- OracleDataReader Members

**GetEnumerator**
This method returns an IEnumerator that can be used to iterate through the collection (record set).

**Declaration**
// ADO.NET 2.0: C#
public override IEnumerator GetEnumerator();

**Return Value**
An IEnumerator that can be used to iterate through the collection (record set).

**Exceptions**
InvalidOperationException - The reader is closed.

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

**GetFieldType**
This method returns the type of the specified column.
Declaration

// ADO.NET 2.0: C#
public override 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, or the specified column is a UDT but no registered custom type mapping exists for the UDT.

IndexOutOfRangeException - The column index is invalid.

Remarks

GetFieldType returns a type that corresponds to the value that the application obtains after invoking the GetValue accessor or Item property on the OracleDataReader. For example, if the column is a string, this method returns a .NET Type object for a .NET string.

If the attribute is a UDT, this method may return either of the following:

- A .NET Type of the custom type if a custom type mapping exists for the Oracle object or collection.

- A .NET Type of string if the column is an Oracle REF.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

GetFloat

This method returns the float value of the specified NUMBER column or BINARY_FLOAT column.

Declaration

// ADO.NET 2.0: C#
public override float GetFloat(int index);

Parameters

- index
  The zero-based column index.

Return Value

The float value of the column.
**Implements**
IDataRecord

**Exceptions**
- InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.
- IndexOutOfRangeException - The column index is invalid.
- InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

**Remarks**
IsDBNull should be called to check for NULL values before calling this method.
Starting with Oracle Database 10g, GetFloat now supports retrieval of data from BINARY_FLOAT columns.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

**GetGuid**
This method is not supported.

**Declaration**
```csharp
// ADO.NET 2.0: C#
public override Guid GetGuid(int index);
```

**Parameters**
- **index**
  The zero-based column index.

**Implements**
IDataRecord

**Exceptions**
- NotSupportedException - This property is not supported.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

**GetInt16**
This method returns the Int16 value of the specified NUMBER column.

**Note:** short is equivalent to Int16.
Declaration

// ADO.NET 2.0: C#
public override short GetInt16(int index);

Parameters

- **index**
  
  The zero-based column index.

Return Value

The Int16 value of the column.

Implements

IDataRecord

Exceptions

- InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.
- IndexOutOfRangeException - The column index is invalid.
- InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

GetInt32

This method returns the Int32 value of the specified NUMBER column.

Note: int is equivalent to Int32.

Declaration

// ADO.NET 2.0: C#
public override 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 1-4
- OracleDataReader Class
- OracleDataReader Members

GetInt64

This method returns the Int64 value of the specified NUMBER column.

----------

Note: long is equivalent to Int64.

----------

Declaration

// ADO.NET 2.0: C#
public override 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.
GetName

This method returns the name of the specified column.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override 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 1-4
- OracleDataReader Class
- OracleDataReader Members

GetOracleBFile

This method returns an `OracleBFile` object of the specified `BFILE` column.

**Declaration**

```csharp
// 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 1-4
- 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
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 1-4
- 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 database 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.

/* Database Setup, if you have not done so yet.
connect scott/tiger@oracle
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,byteCodes) values
(1,'KING','SOFTWARE ENGR', '5657');
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(2,'SCOTT','MANAGER', '5960');
commit;
*/

// 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();
        }
    }
}
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 database 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 GetOracleBlobForUpdate() can have a different value than the OracleBlob obtained from GetOracleBlob() 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.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-83
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.

TheWAIT 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 5-188 for a code example demonstrating usage.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-83

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

Declaration

// C#
public OracleClob GetOracleClob(int index);

Parameters

- index
  The zero-based column index.

Return Value
The OracleClob value of the column.

Exceptions

- InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.
- IndexOutOfRangeException - The column index is invalid.
InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks
IsDBNull should be called to check for NULL values before calling this method.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-83

GetOracleClobForUpdate

GetOracleClobForUpdate returns an updatable OracleClob object of the specified CLOB column.

Overload List:
- GetOracleClobForUpdate(int)
  This method returns an updatable OracleClob object of the specified CLOB column.
- GetOracleClobForUpdate(int, int)
  This method returns an updatable OracleClob object of the specified CLOB column using a WAIT clause.

GetOracleClobForUpdate(int)

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

Declaration
// C#
public OracleClob GetOracleClobForUpdate(int index);

Parameters
- index

The zero-based column index.

Return Value
An updatable OracleClob.

Exceptions
- InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.
- IndexOutOfRangeException - The column index is invalid.
- InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.
Remarks
When the `OracleCommand`'s `ExecuteReader()` method is invoked, all the data fetched by the `OracleDataReader` is from a particular snapshot. Therefore, calling an accessor method on the same column always returns the same value. However, the `GetOracleClobForUpdate()` method incurs a database 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
/* Database Setup, if you have not done so yet.  
connect scott/tiger@oracle  
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,byteCodes) values  
(1,'KING', 'SOFTWARE ENGR', '5657');  
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values  
(2,'SCOTT', 'MANAGER', '5960');  
commit;  
*/  
// 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 1-4
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-83

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 database 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 5-193 for a code example demonstrating usage.
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 1-4
- `OracleDataReader Class`
- `OracleDataReader Members`
- "LOB Support" on page 3-83

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.
OracleDataReader Class

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 1-4
- OracleDataReader Class
- OracleDataReader Members

GetOracleIntervalDS
This method returns an OracleIntervalDS structure of the specified INTERVAL column.

Declaration
// C#
public OracleIntervalDS GetOracleIntervalDS(int index);

Parameters
- index
  The zero-based column index.

Return Value
The OracleIntervalDS value of the column.

Exceptions
InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.
IndexOutOfRangeException - The column index is invalid.
InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks
IsDBNull should be called to check for NULL values before calling this method.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
■ OracleDataReader Class
■ OracleDataReader Members

GetOracleRef

This method returns an OracleRef object of the specified REF column.

Declaration
// C#
public OracleRef GetOracleRef(int index);

Parameters
■ index

The zero-based column index.

Return Value
The OracleRef object of the specified column.

Exceptions
InvalidOperationException - The connection is closed, the reader is closed, the Read method has not been called, or all rows have been read.
IndexOutOfRangeException - The column index is invalid.
InvalidOperationException - The accessor method is invalid for this column type.

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

GetOracleString
This method returns an OracleString structure of the specified column.

Declaration
// C#
public OracleString GetOracleString(int index);

Parameters
- index
  The zero-based column index.

Return Value
The OracleString value of the column.

Exceptions
InvalidOperationException - The connection is closed, the reader is closed, Read() has not been called, or all rows have been read.
IndexOutOfRangeException - The column index is invalid.
InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks
If the column is an Oracle REF column, the string returned is a hexadecimal value that represents the REF in the database.

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

GetOracleTimeStamp
This method returns an OracleTimeStamp structure of the Oracle TimeStamp column.

Declaration
// C#
public OracleTimeStamp GetOracleTimeStamp(int index);
Parameters

<table>
<thead>
<tr>
<th>index</th>
</tr>
</thead>
</table>
| 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.
- **OutOfRange.RangeException** - 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 1-4
- OracleDataReader Class
- OracleDataReader Members

GetOracleTimeStampLTZ

This method returns an OracleTimeStampLTZ structure of the specified Oracle TimeStamp WITH LOCAL TIME ZONE column.

Declaration

```csharp
// C#
public OracleTimeStampLTZ GetOracleTimeStampLTZ(int index);
```

Parameters

<table>
<thead>
<tr>
<th>index</th>
</tr>
</thead>
</table>
| 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.
- **OutOfRange.RangeException** - 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 1-4
- 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 1-4
- OracleDataReader Class
- OracleDataReader Members

GetOracleXmlType
This method returns an OracleXmlType object of the specified XMLType column.

Declaration
// C#
public OracleXmlType GetOracleXmlType(int index);

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
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 later.

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

GetOracleValue

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

Declaration

```csharp
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.

Remarks
If the column is an Oracle object or Oracle collection column and a custom type mapping exists, then a custom type is returned.
If the column is an Oracle REF column, then an OracleRef is returned.
GetOracleValues

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

Declaration

```c#
public int GetOracleValues(object[] values);
```

Parameters

- `values`
  
  An array of objects to hold the ODP.NET types as the column values.

Return Value

The number of ODP.NET types in the `values` array.

Exceptions

- `InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.

Remarks

This method provides a way to retrieve all column values rather than retrieving each column value individually.

The number of column values retrieved is the minimum of the length of the `values` array and the number of columns in the result set.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
- "LOB Support" on page 3-83

GetOrdinal

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

Declaration

```c#
public override 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 1-4
- OracleDataReader Class
- OracleDataReader Members

GetProviderSpecificFieldType
This method returns the provider-specific type of the specified column.

Declaration
// ADO.NET 2.0: C#
public override Type GetProviderSpecificFieldType(int index);

Parameters
- index
  A zero-based column index.

Return Value
The provider-specific type of the specified column. This is a member of the Oracle.DataAccess.Types namespace.

Exceptions
IndexOutOfRangeException - The column index is invalid.
InvalidOperationException - The reader is closed, or the specified column is a UDT but no registered custom type mapping exists for the UDT.

Remarks
GetProviderSpecificFieldType returns a type that corresponds to the value the application obtains after invoking the GetProviderSpecificValue accessor on the OracleDataReader. For example, if the column is a string, this method returns a .NET Type object for an OracleString.

If the attribute is a UDT, this method may return any of the following:
- A .NET Type of the custom type, if the column is an Oracle object or Oracle collection column and a custom type mapping exists.
A .NET Type of `OracleRef` if the column is an Oracle REF.

**GetProviderSpecificValue**

This method returns an object that represents the underlying provider-specific value of the specified ordinal.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override object GetProviderSpecificValue (int index);
```

**Parameters**

`index`

A zero-based column index.

**Return Value**

An `Object` that is a representation of the underlying provider-specific field type.

**Exceptions**

- `IndexOutOfRangeException` - The column index is invalid.
- `InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called or all rows have been read.

**Remarks**

If the column is an Oracle object or collection column, and a custom type mapping exists, a custom type is returned.

If the column is an Oracle REF column, an `OracleRef` is returned.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleDataReader Class`
- `OracleDataReader Members`

**GetProviderSpecificValues**

This method returns an array of objects that represent the underlying provider-specific values.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override int GetProviderSpecificValues(object [ ] values);
```
Parameters

values

An array of objects.

Return Value

The number of Object instances in the array.

Exceptions

InvalidOperationException - The reader is closed.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

GetSchemaTable

This method returns a DataTable that describes the column metadata of the OracleDataReader.

Declaration

// ADO.NET 2.0: C#
public override 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

The OracleDataReader.GetSchemaTable method 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>ColumnName</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></td>
<td></td>
<td>See &quot;IsByteSemantic&quot; on page 5-208 for more information.</td>
</tr>
<tr>
<td>NumericPrecision</td>
<td>System.Int16</td>
<td>The maximum precision of the column, if the column is a numeric data type.</td>
</tr>
<tr>
<td>NumericScale</td>
<td>System.Int16</td>
<td>The scale of the column.</td>
</tr>
<tr>
<td>IsUnique</td>
<td>System.Boolean</td>
<td>Indicates whether or not the column is unique.</td>
</tr>
</tbody>
</table>

This column has valid values for Oracle NUMBER, Oracle INTERVAL YEAR TO MONTH, and Oracle INTERVAL DAY TO SECOND columns. For all other columns, the value is null.

This column has valid values for Oracle NUMBER, Oracle INTERVAL DAY TO SECOND, and the Oracle TIMESTAMP columns. For all other columns, the value is null.

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.
IsKey
System.Boolean
Indicates whether or not the column is a key column.
true if the column is one of a set of columns in the rowset that, taken together, uniquely identify the row.
The set of columns with IsKey set to true must uniquely identify a row in the rowset. There is no requirement that this set of columns is a minimal set of columns.

This set of columns can be generated from one of the following in descending order of priority:
- A base table primary key.
- Any of the unique constraints or unique indexes with the following condition: A NOT NULL constraint must be defined on the column or on all of the columns, in the case of a composite unique constraint or composite unique index.
- Any of the composite unique constraints or composite unique indexes with the following condition: A NULL constraint must be defined on at least one, but not all, of the columns.

An explicitly selected ROWID. false if the column is not required to uniquely identify the row. The value of this property is the same for each occurrence of the base table column in the select list.

IsRowID
System.Boolean
true if the column is a ROWID, otherwise false.

BaseColumnName
System.String
The name of the column in the database if an alias is used for the column.

BaseSchemaName
System.String
The name of the schema in the database that contains the column.

BaseTableName
System.String
The name of the table or view in the database that contains the column.

DataType
System.RuntimeType
Maps to the common language runtime type.

ProviderType
Oracle.DataAccess.Client.OracleDbType
The database column type (OracleDbType) of the column.

AllowDBNull
System.Boolean
true if null values are allowed, otherwise false.

IsAliased
System.Boolean
true if the column is an alias; otherwise false.

IsByteSemantic
System.Boolean
IsByteSemantic is:
true if the ColumnSize value uses bytes semantics
false if ColumnSize uses character semantics
This value is always true when connected to a database version earlier than Oracle9i.

IsExpression
System.Boolean
true if the column is an expression; otherwise false.

IsHidden
System.Boolean
true if the column is hidden; otherwise false.

IsReadOnly
System.Boolean
true if the column is read-only; otherwise false.

IsLong
System.Boolean
true if the column is a LONG, LONG RAW, BLOB, CLOB, or BFILE; otherwise false.

UdtTypeName
System.String
The type name of the UDT.

Table 5–49 (Cont.) OracleDataReader SchemaTable

<table>
<thead>
<tr>
<th>Name</th>
<th>Name Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsKey</td>
<td>System.Boolean</td>
<td>Indicates whether or not 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.</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>
<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: true if the ColumnSize value uses bytes semantics false if ColumnSize uses character semantics This value is always true when connected to a database version earlier than Oracle9i.</td>
</tr>
<tr>
<td>IsExpression</td>
<td>System.Boolean</td>
<td>true if the column is an expression; otherwise false.</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>
<tr>
<td>UdtTypeName</td>
<td>System.String</td>
<td>The type name of the UDT.</td>
</tr>
</tbody>
</table>
Example

This example creates and uses the SchemaTable from the reader.

/* Database Setup, if you have not done so yet.
connect scott/tiger@oracle
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,byteCodes) values
(1,'KING','SOFTWARE ENGR', '5657');
Insert into empInfo(EMPNO,EMPNAME,JOBDESCRIPTION,byteCodes) values
(2,'SCOTT','MANAGER', '5960');
commit;
*/

// C#
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];

    //print out the column info
    Console.WriteLine("Column name: " + row["COLUMNNAME"]);
    Console.WriteLine("Precision: " + row["NUMERICPRECISION"]);
    Console.WriteLine("Scale: " + row["NUMERICSCALE"]);
    reader.Close();

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

This method returns the string value of the specified column.

Declaration

```csharp
// ADO.NET 2.0: C#
public override 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

Call the IsDBNull method to check for null values before calling this method.

If the column is an Oracle REF column, the string returned is a hexadecimal string that represents the REF in the database.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members

GetTimeSpan

This method returns the TimeSpan value of the specified INTERVAL DAY TO SECOND column.

Declaration

```csharp
// C#
public TimeSpan GetTimeSpan(int index);
```
Parameters

- *index*
  
The zero-based column index.

Return Value

The `TimeSpan` value of the column.

Implements

`IDataRecord`

Exceptions

- `InvalidOperationException` - The connection is closed, the reader is closed, `Read()` has not been called, or all rows have been read.
- `IndexOutOfRangeException` - The column index is invalid.
- `InvalidCastException` - The accessor method is invalid for this column type or the column value is `NULL`.

Remarks

`IsDBNull` should be called to check for `NULL` values before calling this method.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleDataReader Class`
- `OracleDataReader Members`

**GetValue**

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

Declaration

```csharp
// ADO.NET 2.0: C#
public override 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, all rows have been read, or no valid custom type mapping has been specified for the Oracle Object or Oracle Collection column.
- `IndexOutOfRangeException` - The column index is invalid.
Remarks
If the column is an Oracle Object or an Oracle Collection column, the .NET custom type corresponding to the custom type mapping is returned.
If the column is an Oracle REF column, a hexadecimal value is returned as a .NET string that represents the REF in the database.
If the UDT is NULL, DBNull.Value is returned

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

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

Declaration
// ADO.NET 2.0: C#
public override 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 1-4
- OracleDataReader Class
- OracleDataReader Members

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

// C#
public XmlReader GetXmlReader(int index);

Parameters

- **index**
  
  The zero-based column index.

Return Value

A .NET XmlTextReader.

Exceptions

InvalidCastException - The accessor method is invalid for this column type or the column value is NULL.

Remarks

IsDBNull should be called to check for NULL values before calling this method.

See Also:

- “Oracle.DataAccess.Client Namespace” on page 1-4
- OracleDataReader Class
- OracleDataReader Members

IsDBNull

This method indicates whether or not the column value is NULL.

Declaration

// ADO.NET 2.0: C#
public override 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 5-152.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleDataReader` Class
- `OracleDataReader` Members

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

Declaration
```csharp
// ADO.NET 2.0: C#
public override 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 1-4
- `OracleDataReader` Class
- `OracleDataReader` Members

Read
This method reads the next row in the result set.

Declaration
```csharp
// ADO.NET 2.0: C#
public override 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 5-152.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataReader Class
- OracleDataReader Members
The `OracleError` class represents an error reported by Oracle.

**Class Inheritance**

```
System.Object
```

**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.

If there are multiple errors, ODP.NET only returns the first error message on the stack.

**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)
        {
            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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleError Members
- OracleError Static Methods
- OracleError Properties
- OracleError Methods
OracleError Members

OracleError members are listed in the following tables.

OracleError Static Methods
The OracleError static method is listed in Table 5–50.

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

OracleError Properties
OracleError properties are listed in Table 5–51.

<table>
<thead>
<tr>
<th>Property</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 5–52.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.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 1-4
- OracleError Class
OracleError Static Methods

The OracleError static method is listed in Table 5–53.

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleError Class
- OracleError Members
OracleError Properties

OracleError properties are listed in Table 5–54.

Table 5–54  OracleError Properties

<table>
<thead>
<tr>
<th>Property</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 1-4
- 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.
DataSource

This property specifies the Oracle service name (TNS name) that identifies the Oracle database.

Declaration

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

Property Value

A string.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleError Class
- OracleError Members
- "Array Binding" on page 3-53

Message

This property specifies the message describing the error.

Declaration

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

Property Value

A string.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleError Class
- OracleError Members

Number

This property specifies the Oracle error number.

Declaration

```csharp
// C#
public int Number {get;}
```

Property Value

An int.
OracleError Properties

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

Procedure

This property specifies the stored procedure that causes the error.

Declaration

```
// C#
public string Procedure {get;}
```

Property Value

The stored procedure name.

Remarks

Represents the stored procedure which creates this OracleError object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
- OracleError Class
- OracleError Members
OracleError Methods

OracleError methods are listed in Table 5–55.

Table 5–55  OracleError Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.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 1-4
- OracleError Class
- OracleError Members

ToString

Overrides Object
This method returns a string representation of the OracleError.

Declaration

// C#
public override string ToString();

Return Value

Returns a string with the format Ora-error number: Class.Method name error message stack trace information.

Example

ORA-24333: zero iteration count

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleError Class
- OracleError Members
OracleErrorCollection Class

An OracleErrorCollection class represents a collection of all errors that are thrown by the Oracle Data Provider for .NET.

Class Inheritance
System.Object
    System.ArrayList

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.
If there are multiple errors, ODP.NET only returns the first error message on the stack.

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);
            }
        }
    }
}
OracleErrorCollection Class

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 5–56.

Table 5–56  OracleErrorCollection Static Methods

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

OracleErrorCollection Properties
OracleErrorCollection properties are listed in Table 5–57.

Table 5–57  OracleErrorCollection Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
<tr>
<td>Count</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
<tr>
<td>IsSynchronized</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
<tr>
<td>Item</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
</tbody>
</table>

OracleErrorCollection Public Methods
OracleErrorCollection public methods are listed in Table 5–58.

Table 5–58  OracleErrorCollection Public Methods

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

See Also:
- “Oracle.DataAccess.Client Namespace” on page 1-4
- OracleErrorCollection Class
OracleErrorCollection Static Methods

The OracleErrorCollection static method is listed in Table 5–59.

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleErrorCollection Class
- OracleErrorCollection Members
OracleErrorCollection Properties

OracleErrorCollection properties are listed in Table 5–60.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
<tr>
<td>Count</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
<tr>
<td>IsSynchronized</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
<tr>
<td>Item</td>
<td>Inherited from System.Collections.ArrayList</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleErrorCollection Class
- OracleErrorCollection Members
OracleErrorCollection Public Methods

OracleErrorCollection public methods are listed in Table 5–61.

Table 5–61  OracleErrorCollection Public Methods

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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
System.Object
    System.Exception
        System.SystemException
            System.Runtime.InteropServices.ExternalException (ADO.NET 2.0 only)
                System.Data.Common.DbException (ADO.NET 2.0 only)

Declaration
// C#
public sealed class OracleException : SystemException

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

Remarks
If there are multiple errors, ODP.NET only returns the first error message on the stack.

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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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
The OracleException static method is listed in Table 5–62.

Table 5–62 OracleException Static Method

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

OracleException Properties
OracleException properties are listed in Table 5–63.

Table 5–63 OracleException Properties

<table>
<thead>
<tr>
<th>Property</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 System.Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from System.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 System.Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from System.Exception</td>
</tr>
</tbody>
</table>

OracleException Methods
OracleException methods are listed in Table 5–64.

Table 5–64 OracleException Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from System.Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.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 System.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 1-4
- OracleException Class
OracleException Static Methods

The `OracleException` static method is listed in Table 5–65.

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleException Class`
- `OracleException Members`
OracleException Properties

OracleException properties are listed in Table 5–66.

<table>
<thead>
<tr>
<th>Property</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 System.Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from System.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 System.Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from System.Exception</td>
</tr>
</tbody>
</table>

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

DataSource

This property specifies the TNS name that contains the information for connecting to an Oracle instance.

Declaration

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

Property Value

The TNS name containing the connect information.

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

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

Procedure

This property specifies the stored procedure that caused the exception.

Declaration

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

Property Value

The stored procedure name.

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

The name of the data provider.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleException Class
- OracleException Members
OracleException Methods

OracleException methods are listed in Table 5–67.

### Table 5–67  OracleException Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from System.Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.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 System.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 1-4
- 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 1-4
- 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 1-4
- 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
System.Object
   System.EventArgs

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 " +
            "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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleInfoMessageEventArgs Members
- OracleInfoMessageEventArgs Static Methods
- OracleInfoMessageEventArgs Properties
- OracleInfoMessageEventArgs Public Methods
- "OracleConnection Class" on page 5-64
OracleInfoMessageEventArgs Members

OracleInfoMessageEventArgs members are listed in the following tables.

OracleInfoMessageEventArgs Static Methods
The OracleInfoMessageEventArgs static methods is listed in Table 5–68.

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

OracleInfoMessageEventArgs Properties
The OracleInfoMessageEventArgs properties are listed in Table 5–69.

<table>
<thead>
<tr>
<th>Property</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 5–70.

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleInfoMessageEventArgs Class
OracleInfoMessageEventArgs Static Methods

The OracleInfoMessageEventArgs static method is listed in Table 5–71.

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members
OracleInfoMessageEventArgs Properties

The OracleInfoMessageEventArgs properties are listed in Table 5-72.

<table>
<thead>
<tr>
<th>Property</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>

### 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 1-4
- 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 1-4
- 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 1-4
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members
OracleInfoMessageEventArgs Public Methods

The OracleInfoMessageEventArgs methods are listed in Table 5–73.

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

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleInfoMessageEventArgs Class
- OracleInfoMessageEventArgs Members
OracleInfoMessageEventHandler Delegate

The OracleInfoMessageEventHandler represents the signature of the method that handles the OracleConnection.InfoMessage event.

Declaration

```csharp
// C#
public delegate void OracleInfoMessageEventHandler(object sender,
    OracleInfoMessageEventArgs eventArgs);
```

Parameters

- **sender**
  - The source of the event.
- **eventArgs**
  - The OracleInfoMessageEventArgs object that contains the event data.

Requirements

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "InfoMessage" on page 5-114
OracleParameter Class

An OracleParameter object represents a parameter for an OracleCommand or a DataSet column.

Class Inheritance
System.Object
    System.MarshalByRefObject
        System.Data.Common.DbParameter (ADO.NET 2.0 only)

Declaration
// ADO.NET 2.0: C#
public sealed class OracleParameter : DbParameter, 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 ="Your SQL command goes here";

        // Execute the command
        cmd.ExecuteNonQuery();

        // Retrieve the result
        OracleDataReader rdr = cmd.ExecuteReader();
        while (rdr.Read())
        {
            Console.WriteLine("Empno: "+rdr["empno"]);
        }
    }
}
OracleParameter Class

"insert into emp(empno, ename, deptno) values(:1, :2, :3);"

cmd.ExecuteNonQuery();

Console.WriteLine("Record for employee id {0} has been inserted.", maxno + 10);
}
}

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 5–74.

**Table 5–74 OracleParameter Constructors**

<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 5–75.

**Table 5–75 OracleParameter Static Methods**

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

**OracleParameter Properties**
OracleParameter properties are listed in Table 5–76.

**Table 5–76 OracleParameter Properties**

<table>
<thead>
<tr>
<th>Property</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 or not the OracleParameter represents a collection, and if so, specifies the collection type</td>
</tr>
<tr>
<td>DbType</td>
<td>Specifies the data type 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>Not supported</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 data type</td>
</tr>
<tr>
<td>OracleDbTypeEx</td>
<td>Specifies the Oracle data type to bind the parameter as, but returns a .NET type as output</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>
Table 5–76 (Cont.) OracleParameter Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Specifies the maximum size, in bytes or characters, of the data transmitted to or from the database. 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>SourceColumnNullableMapping</td>
<td>Specifies a value which indicates whether the source column is nullable Supported Only in ADO.NET 2.0-Compliant ODP.NET</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>UdtTypeName</td>
<td>Specifies the Oracle user-defined type name if the parameter is a user-defined data type</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the value of the Parameter</td>
</tr>
</tbody>
</table>

OracleParameter Public Methods

OracleParameter public methods are listed in Table 5–77.

Table 5–77 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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases allocated resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>ResetDbType</td>
<td>Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>ResetOracleDbType</td>
<td>Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 data type.

- **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, data type, and parameter direction.

- **OracleParameter(string, OracleDbType, object, ParameterDirection)**
  This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, value, and direction.

- **OracleParameter(string, OracleDbType, int)**
  This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, and size.

- **OracleParameter(string, OracleDbType, int, string)**
  This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, 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, data type, 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, data type, size, value, and direction.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362
- "OracleParameterCollection Class" on page 5-283

OracleParameter (string, OracleDbType)

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

Declaration

// C#
public OracleParameter(string parameterName, OracleDbType oraType);

Parameters

- parameterName
  
  The parameter name.

- oraType
  
  The data type 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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362
- "OracleParameterCollection Class" on page 5-283

### OracleParameter(string, object)

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

**Declaration**
```c#
public OracleParameter(string parameterName, object obj);
```

**Parameters**
- `parameterName`
  The parameter name.
- `obj`
  The 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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362
- "OracleParameterCollection Class" on page 5-283

OracleParameter(string, OracleDbType, ParameterDirection)
This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, and parameter direction.

Declaration

```csharp
public OracleParameter(string parameterName, OracleDbType type,
                        ParameterDirection direction);
```

Parameters

- `parameterName`
  The parameter name.
- `type`
  The data type of the OracleParameter.
- `direction`
  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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362
- "OracleParameterCollection Class" on page 5-283

OracleParameter(string, OracleDbType, object, ParameterDirection)

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

Declaration

```csharp
// C#
public OracleParameter(string parameterName, OracleDbType type, object obj,
   ParameterDirection direction);
```

Parameters

- **parameterName**
  
  The parameter name.

- **type**
  
  The data type of the OracleParameter.

- **obj**
  
  The value of the OracleParameter.

- **direction**
  
  The ParameterDirection value.

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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362
- "OracleParameterValueCollection Class" on page 5-283

**OracleParameter(string, OracleDbType, int)**

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, and size.

**Declaration**

```csharp
// C#
public OracleParameter(string parameterName, OracleDbType type,
                        int size);
```

**Parameters**

- `parameterName`
  The parameter name.
- `type`
  The data type of the OracleParameter.
- `size`
  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
- DbType - 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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362
- "OracleParameterCollection Class" on page 5-283

OracleParameter(string, OracleDbType, int, string)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, size, and source column.

Declaration

// C#
public OracleParameter(string parameterName, OracleDbType type, int size,
        string srcColumn);

Parameters

- parameterName
  The parameter name.
- type
  The data type of the OracleParameter.
- size
  The size of the OracleParameter value.
- srcColumn
  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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362

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, data type, 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
  The parameter name.
- oraType
  The data type of the OracleParameter.
- size
  The size of the OracleParameter value.
- direction
  The ParameterDirection value.
- `isNullble`
  An indicator that specifies if the parameter value can be null.

- `precision`
  The precision of the parameter value.

- `scale`
  The scale of the parameter value.

- `srcColumn`
  The name of the source column.

- `srcVersion`
  The `DataRowVersion` value.

- `obj`
  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`
- `isNullble` - `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 1-4
- `OracleParameter Class`
- `OracleParameter Members`
- "OracleParameterStatus Enumeration" on page 5-362
OracleParameter(string, OracleDbType, int, object, ParameterDirection)

This constructor instantiates a new instance of the OracleParameter class using the supplied parameter name, data type, size, value, and direction.

Declaration

// C#
public OracleParameter(string parameterName, OracleDbType type, int size, object obj, ParameterDirection direction);

Parameters

- **parameterName**
  The parameter name.
- **type**
  The data type of the OracleParameter.
- **size**
  The size of the OracleParameter value.
- **obj**
  The value of the OracleParameter.
- **direction**
  The ParameterDirection value.

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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362
- "OracleParameterCollection Class" on page 5-283
OracleParameter Static Methods

The `OracleParameter` static method is listed in Table 5–78.

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleParameter` Class
- `OracleParameter` Members
OracleParameter properties are listed in Table 5–79.

### Table 5–79  OracleParameter Properties

<table>
<thead>
<tr>
<th>Property</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 or not the OracleParameter represents a collection, and if so, specifies the collection type</td>
</tr>
<tr>
<td>DbType</td>
<td>Specifies the data type 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>Not supported</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 data type</td>
</tr>
<tr>
<td>OracleDbTypeEx</td>
<td>Specifies the Oracle data type to bind the parameter as, but returns a .NET type as output</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 database. 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>
</tbody>
</table>
| SourceColumnNullMapping| Specifies a value which indicates whether the source column is nullable  
*Supported Only in ADO.NET 2.0-Compliant ODP.NET*                                         |
| SourceVersion          | Specifies the DataRowVersion value to use when loading the Value property of the parameter                                                  |
| Status                 | Indicates the status of the execution related to the data in the Value property                                                            |
| UdtTypeName            | Specifies the Oracle user-defined type name if the parameter is a user-defined data type                                                   |
| Value                  | Specifies the value of the Parameter                                                                                                       |
ArrayBindSize

This property specifies the maximum size, in bytes or characters, of the data for each array element transmitted to or from the database. 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)";

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();
}
}
**Property Value**

An array of **OracleParameterStatus** enumerated values.

**Exceptions**

ArgumentOutOfRangeException - The Status value specified is invalid.

**Remarks**

Default = null.

*ArrayBindStatus* is used for Array Bind and PL/SQL Associative Array execution only.

Before execution, *ArrayBindStatus* indicates the bind status of each element in the *Value* property. After execution, it contains the execution status of each element in the *Value* property.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- **OracleParameter Class**
- **OracleParameter Members**
- "ArrayBindCount" on page 5-13
- "OracleParameterStatus Enumeration" on page 5-362
- "Value" on page 5-277 for more information on binding Associative Arrays
- "ArrayBindSize" on page 5-265

**CollectionType**

This property specifies whether or not 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 1-4
- **OracleParameter Class**
- **OracleParameter Members**
DbType

This property specifies the data type of the parameter using the Data.DbType enumeration type.

Declaration
// ADO.NET 2.0: C#
public override DbType DbType {get; set; }

PropertyValue
A DbType enumerated value.

Implements
IDataParameter

Exceptions
ArgumentException - The DbType value specified is invalid.

Remarks
Default = DbType.String

DbType is the data type 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 1-4
■ OracleParameter Class
■ OracleParameter Members
■ "Inference of OracleDbType from DbType” on page 3-48
■ "CollectionType” on page 5-267

Direction

This property specifies whether the parameter is input-only, output-only, bi-directional, or a stored function return value parameter.

Declaration
// ADO.NET 2.0: C#
public override ParameterDirection Direction { get; set; }

PropertyValue
A ParameterDirection enumerated value.

Implements
IDataParameter

Exceptions
ArgumentOutOfRangeException - The ParameterDirection value specified is invalid.
OracleParameter Class

Remarks
Default = ParameterDirection.Input
Possible values: Input, InputOutput, Output, and ReturnValue.

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

IsNullable
This property is not supported.

Declaration
// ADO.NET 2.0: C#
public override bool IsNullable { get; set; }

Implements
IDataParameter

Property Value
This property is not supported.

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

- `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 1-4
- OracleParameter Class
- OracleParameter Members

**OracleDbType**

This property specifies the Oracle data type.

**Declaration**

```csharp
// 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 data type 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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleDbType Enumeration" on page 5-360
- "Inference of DbType from OracleDbType" on page 3-47
- "CollectionType" on page 5-267

**OracleDbTypeEx**

This property specifies the Oracle data type to bind the parameter as, but returns a .NET type as output.
Declaration

// C#
public OracleDbType OracleDbTypeEx { get; set; }

Property Value
An OracleDbType enumerated value.

Remarks
This property is used by applications that need to bind a parameter value as an Oracle type, but need a .NET type back for output. This property should be used with an output or input/output parameter. For an input parameter, using OracleDbTypeEx has the same affect as using OracleDbType. The .NET type that is returned for the output is the .NET type that the Oracle type closely maps to.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleDbType Enumeration" on page 5-360
- "OracleDbType" on page 5-270
- "Inference of DbType from OracleDbType" on page 3-47
- "CollectionType" on page 5-267

ParameterName
This property specifies the name of the parameter.

Declaration

// ADO.NET 2.0: C#
public override 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 1-4
- OracleParameter Class
- OracleParameter Members
**Precision**

This property specifies the maximum number of digits used to represent the Value property.

**Declaration**

```csharp
// 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 1-4
- OracleParameter Class
- OracleParameter Members
- "Value" on page 5-277

**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 1-4
- OracleParameter Class
- OracleParameter Members
- "Value" on page 5-277

Size

This property specifies the maximum size, in bytes or characters, of the data transmitted to or from the database.

Declaration

// ADO.NET 2.0: C#
public override int Size { get; set; }

Property Value

int

Exceptions

ArgumentException - 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 data types: ignored
OracleParameter Properties

- Variable length data types: describes the maximum amount of data transmitted to or from the database. 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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleDbType Enumeration" on page 5-360
- "CollectionType" on page 5-267
- "ArrayBindSize" on page 5-265
- "ArrayBindStatus" on page 5-266
- "Value" on page 5-277

**SourceColumn**

This property specifies the name of the DataTable Column of the DataSet.

**Declaration**

// ADO.NET 2.0: C#
public override string SourceColumn { get; set; }

**Property Value**

A string.

**Implements**

IDataParameter

**Remarks**

Default = empty string
SourceColumnNullMapping
This property specifies a value which indicates whether the source column is nullable.

Declaration
// ADO.NET 2.0: C#
public bool SourceColumnNullMapping { get; set; }

Property Value
Returns true if the source column can be nullified; otherwise, returns false.

Remarks
The default value is false.

See Also:
■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleParameter Class
■ OracleParameter Members

SourceVersion
This property specifies the DataRowVersion value to use when loading the Value property of the parameter.

Declaration
// ADO.NET 2.0: C#
public override DataRowVersion SourceVersion { get; set; }

Property Value
DataRowVersion

Implements
IDataParameter

Exceptions
ArgumentOutOfRangeException - The DataRowVersion value specified is invalid.

Remarks
Default = DataRowVersion.Current
SourceVersion is used by the OracleDataAdapter.UpdateCommand() during the OracleDataAdapter.Update to determine whether the original or current value is used for a parameter value. This allows primary keys to be updated. This
property is ignored by the OracleDataAdapter.InsertCommand() and the OracleDataAdapter.DeleteCommand().

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameter Class
- OracleParameter Members

Status

This property indicates the status of the execution related to the data in the Value property.

Declaration

```csharp
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 if:

- 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 1-4
- OracleParameter Class
- OracleParameter Members
- "OracleParameterStatus Enumeration" on page 5-362
- "ArrayBindStatus" on page 5-266

UdtTypeName

This property specifies the Oracle user-defined type name if the parameter is a user-defined data type.
Declaration

// C#
public string UdtTypeName {get; set;}

Property Value

Name of the Oracle UDT.

Remarks

The UdtTypeName property corresponds to the user-defined type name of the parameter. This property must always be specified if the parameter is a user-defined type.

Note that when a custom object is provided as an input parameter value, it is converted to the Oracle UDT that is specified by the custom type mapping on the connection used to execute the command.

The Oracle UDT specified by the custom type mapping and by the OracleParameter.UdtTypeName property differs if the application binds a custom object that represents a subtype of the parameter type.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameter Class
- OracleParameter Members

Value

This property specifies the value of the Parameter.

Declaration

// ADO.NET 2.0: C#
public override 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 3-49
- "ArrayBindCount" on page 5-13
- "ArrayBindSize" on page 5-265
- "ArrayBindStatus" on page 5-266
- "OracleDbType Enumeration" on page 5-360

For input parameters the value is:

- Bound to the OracleCommand that is sent to the database.
- Converted to the data type specified in OracleDbType or DbType when the provider sends the data to the database.

For output parameters the value is:

- Set on completion of the OracleCommand (true for return value parameters also).
- Set to the data from the database, to the data type 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 1-4
- OracleParameter Class
- OracleParameter Members
- "ArrayBindCount" on page 5-13
- "OracleParameterStatus Enumeration" on page 5-362
OracleParameter Public Methods

OracleParameter public methods are listed in Table 5–80.

<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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases allocated resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>ResetDbType</td>
<td>Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter</td>
</tr>
<tr>
<td>ResetOracleDbType</td>
<td>Resets the type associated with the parameter so that it can infer its type from the value passed in the parameter</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameter Class
- OracleParameter Members

**Clone**

This method creates a shallow copy of an OracleParameter object.

**Declaration**

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

**Return Value**

An OracleParameter object.

**Implements**

ICloneable

**Remarks**

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

// C#

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 1-4
- 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 1-4
- OracleParameter Class
- OracleParameter Members
ResetDbType

This method resets the type associated with the parameter so that it can infer its type from the value passed in the parameter.

Declaration

```csharp
// ADO.NET 2.0: C#
public override void ResetDbType();
```

Remarks

If an application does not set the DbType or OracleDbType properties of an OracleParameter object, then these values are inferred from the value set by the application to that OracleParameter object. Calling ResetDbType method resets these properties so that OracleParameter can again infer its type from the value passed into the OracleParameter. Calling this method affects both the DbType and OracleDbType properties of the OracleParameter object.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameter Class
- OracleParameter Members

ResetOracleDbType

This method resets the type associated with the parameter so that it can infer its type from the value passed in the parameter.

Declaration

```csharp
// ADO.NET 2.0: C#
public override void ResetOracleDbType();
```

Remarks

If an application does not set the DbType or OracleDbType properties of an OracleParameter object, then these values are inferred from the value set by the application to that OracleParameter object. Calling the ResetOracleDbType method resets these properties so that OracleParameter can again infer its type from the value passed into the OracleParameter. Calling this method affects both the DbType and OracleDbType properties of the OracleParameter object.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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
System.Object
    System.MarshalByRefObject
        System.Data.Common.DbParameterCollection (ADO.NET 2.0 only)

Declaration
// ADO.NET 2.0: C#
public sealed class OracleParameterCollection : DbParameterCollection,
    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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 5–81.

**Table 5–81 OracleParameterCollection Static Methods**

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

**OracleParameterCollection Properties**

OracleParameterCollection properties are listed in Table 5–82.

**Table 5–82 OracleParameterCollection Properties**

<table>
<thead>
<tr>
<th>Property</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 5–83.

**Table 5–83 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>AddRange</td>
<td>Adds elements to the end of the OracleParameterCollection</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>Clear</td>
<td>Removes all the OracleParameter objects from the collection</td>
</tr>
<tr>
<td>Contains</td>
<td>Indicates whether or not 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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>IndexOf</td>
<td>Returns the index of the objects in the collection (Overloaded)</td>
</tr>
</tbody>
</table>
### OracleParameterCollection Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 System.Object</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
OracleParameterCollection Static Methods

The `OracleParameterCollection` static method is listed in Table 5–84.

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleParameterCollection` Class
- `OracleParameterCollection` Members
OracleParameterCollection Properties

OracleParameterCollection properties are listed in Table 5–85.

Table 5–85 OracleParameterCollection Properties

<table>
<thead>
<tr>
<th>Property</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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

Count

This property specifies the number of OracleParameter objects in the collection.

Declaration

// ADO.NET 2.0: C#
public override int Count {get;}

Property Value

The number of OracleParameter objects.

Implements

ICollection

Remarks

Default = 0

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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.
Item[int]

This property gets and sets the OracleParameter object at the index specified by the supplied parameterIndex.

Declaration

// C#
public object Item[int parameterIndex] {get; set;}

Property Value

An object.

Implements

IList

Exceptions

IndexOutOfRangeException - The supplied index does not exist.

Remarks

The OracleParameterCollection class is a zero-based index.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

Item[string]

This property gets and sets the OracleParameter object using the parameter name specified by the supplied parameterName.

Declaration

// C#
public OracleParameter Item[string parameterName] {get; set;};

Property Value

An OracleParameter.

Implements

IDataParameterCollection

Exceptions

IndexOutOfRangeException - The supplied parameter name does not exist.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members
OracleParameterCollection Public Methods

OracleParameterCollection public methods are listed in Table 5–86.

### Table 5–86 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>AddRange</td>
<td>Adds elements to the end of the OracleParameterCollection</td>
</tr>
<tr>
<td></td>
<td>Supported Only in ADO.NET 2.0-Compliant ODP.NET</td>
</tr>
<tr>
<td>Clear</td>
<td>Removes all the OracleParameter objects from the collection</td>
</tr>
<tr>
<td>Contains</td>
<td>Indicates whether or not 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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.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 System.Object</td>
</tr>
</tbody>
</table>

#### See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

### Add

Add adds objects to the collection.

**Overload List:**
- Add(object)
  - This method adds the supplied object to the collection.
- Add(OracleParameter)
This method adds the supplied OracleParameter object to the collection.

- **Add(string, object)**
  This method adds an OracleParameter object to the collection using the supplied name and object value.

- **Add(string, OracleDbType)**
  This method adds an OracleParameter object to the collection using the supplied name and database type.

- **Add(string, OracleDbType, ParameterDirection)**
  This method adds an OracleParameter object to the collection using the supplied name, database type, and direction.

- **Add(string, OracleDbType, object, ParameterDirection)**
  This method adds an OracleParameter object to the collection using the supplied name, database type, parameter value, and direction.

- **Add(string, OracleDbType, int, object, ParameterDirection)**
  This method adds an OracleParameter object to the collection using the supplied name, database type, size, parameter value, and direction.

- **Add(string, OracleDbType, int)**
  This method adds an OracleParameter object to the collection using the supplied name, database type, and size.

- **Add (string, OracleDbType, int, string)**
  This method adds an OracleParameter object to the collection using the supplied name, database type, size, and source column.

- **Add(string, OracleDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object)**
  This method adds an OracleParameter object to the collection using the supplied name, database type, size, direction, null indicator, precision, scale, source column, source version, and parameter value.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

**Add(object)**

This method adds the supplied object to the collection.

**Declaration**

// ADO.NET 2.0: C#
public override int Add(object obj);

**Parameters**

- **obj**
  The supplied object.
Return Value
The index at which the new OracleParameter is added.

Implements
IList

Remarks
InvalidOperationException - The supplied obj cannot be cast to an OracleParameter object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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
   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 1-4
- 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
   The parameter name.
■ val
   The OracleParameter value.
Return Value
The newly created OracleParameter object which was added to the collection.

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

Add(string, OracleDbType)
This method adds an OracleParameter object to the collection using the supplied name and database type.

Declaration
// C#
public OracleParameter Add(string name, OracleDbType dbType);

Parameters
- name
  The parameter name.
- dbType
  The data type of the OracleParameter.

Return Value
The newly created OracleParameter object which was added to the collection.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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
  The parameter name.
- dbType
  The data type of the OracleParameter.
- direction
  The OracleParameter direction.
Return Value
The newly created OracleParameter object which was added to the collection.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members
- "OracleDbType Enumeration" on page 5-360

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
  The parameter name.
- dbType
  The data type of the OracleParameter.
- val
  The OracleParameter value.
- dir
  The ParameterDirection value.

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);
}
Add(string, OracleDbType, int, object, ParameterDirection)

This method adds an OracleParameter object to the collection using the supplied name, database type, size, parameter value, and direction.

Declaration

// C#
public OracleParameter Add(string name, OracleDbType dbType, int size, object val, ParameterDirection dir);

Parameters

- **name**
  The parameter name.

- **dbType**
  The data type of the OracleParameter.

- **size**
  The size of OracleParameter.

- **val**
  The OracleParameter value.

- **dir**
  The ParameterDirection value.

Return Value

The newly created OracleParameter object which was added to the collection.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members
- "OracleDbType Enumeration" on page 5-360

Add(string, OracleDbType, int)

This method adds an OracleParameter object to the collection using the supplied name, database type, and size.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members
- "OracleDbType Enumeration" on page 5-360
Declaration

// C#
public OracleParameter Add(string name, OracleDbType dbType, int size);

Parameters

- name
  The parameter name.
- dbType
  The data type of the OracleParameter.
- size
  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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

Add (string, OracleDbType, int, string)

This method adds an OracleParameter object to the collection using the supplied name, database type, size, and source column.

Declaration

// C#
public OracleParameter Add(string name, OracleDbType dbType, int size, string srcColumn);
Parameters

- **name**
  The parameter name.

- **dbType**
  The data type of the `OracleParameter`.

- **size**
  The size of `OracleParameter`.

- **srcColumn**
  The name of the source column.

Return Value

An `OracleParameter`.

See Also:

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

```csharp
// 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**
  The parameter name.

- **dbType**
  The data type of the `OracleParameter`.

- **size**
  The size of `OracleParameter`.

- **dir**
  The `ParameterDirection` value.

- **isNullable**
  An indicator that specifies if the parameter value can be null.

- **precision**
  The precision of the parameter value.
- **scale**
  The scale of the parameter value.
- **srcColumn**
  The name of the source column.
- **version**
  The DataRowVersion value.
- **val**
  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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

### AddRange
This method adds elements to the end of the OracleParameterCollection.

#### Declaration
```csharp
// ADO.NET 2.0: C#
public override void AddRange(Array paramArray);
```

#### Parameters
- **paramArray**
  An array of OracleParameter objects.

#### Exceptions
- ArgumentNullException - The input parameter is null.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

### Clear
This method removes all the OracleParameter objects from the collection.

#### Declaration
```csharp
// ADO.NET 2.0: C#
```
public override void Clear();

**Implements**
IList

**Example**

```csharp
// 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.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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

**Contains**

Contains indicates whether or not the supplied object exists in the collection.

**Overload List:**
- Contains(object)
  This method indicates whether or not the supplied object exists in the collection.
- Contains(string)
  This method indicates whether or not an OracleParameter object exists in the collection using the supplied string.
Contains(object)

This method indicates whether or not the supplied object exists in the collection.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override bool Contains(object obj)
```

**Parameters**

- `obj`
  
  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**

```csharp
// 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();
    }
}
```

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleParameterCollection Class`
- `OracleParameterCollection Members`
// 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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

Contains(string)

This method indicates whether or not an OracleParameter object exists in the collection using the supplied string.

Declaration

// ADO.NET 2.0: C#
public override bool Contains(string name);

Parameters

■ name

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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

CopyTo

This method copies OracleParameter objects from the collection, starting with the supplied index to the supplied array.

Declaration

// ADO.NET 2.0: C#
public override void CopyTo(Array array, int index);

Parameters

- array
  The specified array.
- index
  The array index.

Implements

IICollection

See Also:

- "Oracle.DataAccess.Client Namespace” on page 1-4
- 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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

### IndexOf(object)

This method returns the index of the OracleParameter object in the collection.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override int IndexOf(object obj);
```

**Parameters**

- **obj**
  The specified object.

**Return Value**

Returns the index of the OracleParameter object in the collection.

**Implements**

IList

**Exceptions**

InvalidCastException - The supplied `obj` cannot be cast to an OracleParameter object.

**Remarks**

Returns the index of the supplied OracleParameter `obj` in the collection.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

### IndexOf(String)

This method returns the index of the OracleParameter object with the specified name in the collection.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override int IndexOf(String name);
```

**Parameters**

- **name**
  The name of parameter.
OracleParameterCollection Class

Return Value
Returns the index of the supplied OracleParameter in the collection.

Implements
IDataParameterCollection

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

Insert
This method inserts the supplied OracleParameter object to the collection at the specified index.

Declaration
// ADO.NET 2.0: C#
public override void Insert(int index, object obj);

Parameters
- index
  The specified index.
- obj
  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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members

Remove
This method removes the supplied OracleParameter from the collection.

Declaration
// ADO.NET 2.0: C#
public override void Remove(object obj);

Parameters
- obj
  The specified 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.Remove(prm1);

        // 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 1-4
- 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 1-4
  - `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**

```csharp
// ADO.NET 2.0: C#
public override void RemoveAt(int index);
```

**Parameters**

- **index**
  The specified index from which the `OracleParameter` is to be removed.

**Implements**

- `IList`

  **See Also:**
  - "Oracle.DataAccess.Client Namespace" on page 1-4
  - `OracleParameterCollection Class`
  - `OracleParameterCollection Members`

**RemoveAt(String)**

This method removes from the collection the `OracleParameter` object specified by the supplied name.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override 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 1-4
- OracleParameterCollection Class
- OracleParameterCollection Members
OraclePermission Class

An OraclePermission object enables ODP.NET to enforce imperative security and helps ensure that a user has a security level adequate for accessing data.

Class Inheritance
System.Object
        System.Data.Common.DBDataPermission

Declaration
// ADO.NET 2.0: C#
public class OraclePermission: DBDataPermission

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

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OraclePermission Members
- OraclePermission Constructor
- OraclePermission Static Methods
- OraclePermission Public Properties
- OraclePermission Public Methods
OraclePermission Members

OraclePermission members are listed in the following tables.

OraclePermission Constructors
The OraclePermission constructor is listed in Table 5–87.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OraclePermission Constructor</td>
<td>Instantiates a new instance of the OraclePermission class.</td>
</tr>
</tbody>
</table>

OraclePermission Static Methods
The OraclePermission static methods are listed in Table 5–88.

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>RevertAll</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>RevertAssert</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>RevertDeny</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>RevertPermitOnly</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
</tbody>
</table>

OraclePermission Public Properties
The OraclePermission public properties are listed in Table 5–92.

<table>
<thead>
<tr>
<th>Public Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowBlankPassword</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td></td>
<td>OraclePermission does not support this property.</td>
</tr>
</tbody>
</table>

OraclePermission Public Methods
The OraclePermission public methods are listed in Table 5–90.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new connection string fragment and a list of restricted keywords to the OraclePermission object</td>
</tr>
<tr>
<td>Assert</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>Copy</td>
<td>Returns a copy of the current permission object</td>
</tr>
<tr>
<td>Demand</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>Deny</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
</tbody>
</table>
### Table 5–90  (Cont.) OraclePermission Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>FromXml</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Intersect</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td>IsSubsetOf</td>
<td>Returns a boolean value that indicates whether or not the current permission is a subset of the target permission</td>
</tr>
<tr>
<td>IsUnrestricted</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td>PermitOnly</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>ToXml</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td>Union</td>
<td>Inherited from DBDataPermission</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermission Class
OraclePermission Constructor

The OraclePermission constructor instantiates a new instance of the OraclePermission class.

Declaration

```csharp
// C#
public OraclePermission (PermissionState state);
```

Parameters

- `state`

The `state` parameter takes one of the following two values:
  - `PermissionState.None`
  - `PermissionState.Unrestricted`

Exceptions

ArgumentException - The `PermissionState` value is invalid.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermission Class
- OraclePermission Members
OraclePermission Static Methods

The OraclePermission static methods are listed in Table 5–91.

Table 5–91  OraclePermission Static Methods

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>RevertAll</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>RevertAssert</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>RevertDeny</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>RevertPermitOnly</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermission Class
- OraclePermission Members
OraclePermission Public Properties

The OraclePermission public methods are listed in Table 5–92.

Table 5–92  OraclePermission Public Properties

<table>
<thead>
<tr>
<th>Public Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowBlankPassword</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td></td>
<td>OraclePermission ignores the value of this property. Any value set for this property, for an OraclePermission object, is ignored.</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermission Class
- OraclePermission Members
OraclePermission Public Methods

The OraclePermission public methods are listed in Table 5–93.

Table 5–93  OraclePermission Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new connection string fragment and a list of restricted keywords to the OraclePermission object</td>
</tr>
<tr>
<td>Assert</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>Copy</td>
<td>Returns a copy of the current permission object</td>
</tr>
<tr>
<td>Demand</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>Deny</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>FromXml</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Intersect</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td>IsSubsetOf</td>
<td>Returns a boolean value that indicates whether or not the current permission is a subset of the target permission</td>
</tr>
<tr>
<td>IsUnrestricted</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td>PermitOnly</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from CodeAccessPermission</td>
</tr>
<tr>
<td>ToXml</td>
<td>Inherited from DBDataPermission</td>
</tr>
<tr>
<td>Union</td>
<td>Inherited from DBDataPermission</td>
</tr>
</tbody>
</table>

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

Add

This method adds a new connection string fragment and a list of restricted keywords to the OraclePermission object.

Declaration

// C#
public void Add(string connStr, string keyRestrict, KeyRestrictionBehavior behavior);
The connection string fragment.

- **keyRestrict**
  The key restrictions.

- **behavior**
  One of the following `KeyRestrictionBehavior` enumerations:
  - `AllowOnly`
  - `PreventUsage`

**Exceptions**

`ArgumentException` - The `KeyRestrictionBehavior` value or the format of the `connStr` or `keyRestrict` string is invalid.

**Remarks**

The `Add` method configures the connection strings allowed or disallowed by the permission object.

Opening an `OracleConnection` is allowed or denied based upon the connection string fragment, key restrictions combination, and the key restriction behavior.

In the following example, `KeyRestrictionBehavior.AllowOnly` allows connection strings that use `orcl` as the Data Source with any User Id and Password combination but no other connection string keywords. Connection string keywords other than User Id and Password cause security exceptions.

```csharp
orclPermission.Add("Data Source=orcl;","User Id=;Password=;", KeyRestrictionBehavior.AllowOnly);
```

In the next example, `KeyRestrictionBehavior.PreventUsage` restricts connection strings that use the keyword `Pooling`. Use of the `Pooling` keyword causes an exception.

```csharp
orclPermission.Add("Data Source=orcl;","Pooling=;", KeyRestrictionBehavior.PreventUsage);
```

As a general rule, in an unrestricted environment, any connection string that is not allowed is restricted and throws a security exception.

If a connection string fragment contains key-value pairs for the `password` and `proxy password` attributes, then values for these attributes are ignored. However, the presence of the attributes themselves is still checked. This means that the connection is allowed only if the `password` and `proxy password` attributes keywords are allowed in the connection string.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OraclePermission Class`
- `OraclePermission Members`

**Copy**

This method returns a copy of the current permission object.
Declaration

// C#
public override IPermission Copy();

Return Value
A copy of the OraclePermission object.

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

IsSubsetOf

This method returns a boolean value that indicates whether or not the current permission is a subset of the target permission.

Declaration

// C#
public override bool IsSubsetOf(IPermission target);

Parameters
- target
  A permission that must be of type OraclePermission.

Return Value
A bool value that indicates whether or not the current permission is a subset of the target permission.

Exceptions
ArgumentException - The permission is not of the OraclePermission type.

Remarks
The AllowBlankPassword property is ignored when evaluating whether or not the current permission is a subset of the target permission.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermission Class
- OraclePermission Members
OraclePermissionAttribute Class

An OraclePermissionAttribute object enables ODP.NET to enforce declarative security and helps ensure that a user has a security level adequate for accessing data.

Class Inheritance
System.Object
    System.Attribute
                System.Data.Common.DBDataPermissionAttribute

Declaration
// ADO.NET 2.0:C#
public sealed class OraclePermissionAttribute: DBDataPermissionAttribute

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

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermissionAttribute Members
- OraclePermissionAttribute Constructor
- OraclePermissionAttribute Static Methods
- OraclePermissionAttribute Public Properties
- OraclePermissionAttribute Public Methods
OraclePermissionAttribute Members

OraclePermissionAttribute members are listed in the following tables.

OraclePermissionAttribute Constructor
The OraclePermissionAttribute constructor is listed in Table 5–94.

Table 5–94 OraclePermission Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OraclePermissionAttribute</td>
<td>Instantiates a new instance of the OraclePermissionAttribute class.</td>
</tr>
</tbody>
</table>

OraclePermissionAttribute Static Methods
The OraclePermissionAttribute static methods are listed in Table 5–95.

Table 5–95 OraclePermissionAttribute Static Methods

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCustomAttribute</td>
<td>Inherited from System.Attribute (Overloaded)</td>
</tr>
<tr>
<td>GetCustomAttributes</td>
<td>Inherited from System.Attribute (Overloaded)</td>
</tr>
<tr>
<td>IsDefined</td>
<td>Inherited from System.Attribute (Overloaded)</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

OraclePermissionAttribute Public Properties
The OraclePermissionAttribute public properties are listed in Table 5–96.

Table 5–96 OraclePermissionAttribute Public Properties

<table>
<thead>
<tr>
<th>Public Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Inherited from SecurityAttribute</td>
</tr>
<tr>
<td>AllowBlankPassword</td>
<td>Inherited from DBDataPermissionAttribute. OraclePermissionAttribute ignores this property. Any value set for this property, for an OraclePermissionAttribute object, is ignored.</td>
</tr>
<tr>
<td>ConnectionString</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>KeyRestrictionBehavior</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>KeyRestrictions</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>TypeId</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Unrestricted</td>
<td>Inherited from SecurityAttribute</td>
</tr>
</tbody>
</table>

OraclePermissionAttribute Public Methods
The OraclePermissionAttribute public methods are listed in Table 5–97.
### OraclePermissionAttribute Public Methods

<table>
<thead>
<tr>
<th>Public Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatePermission</td>
<td>Returns a new OraclePermissionAttribute object that is configured based on the attributes set</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Match</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ShouldSerializeConnectionString</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>ShouldSerializeKeyRestrictions</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermissionAttribute Class
OraclePermissionAttribute Constructor

The OraclePermissionAttribute constructor instantiates new instances of the OraclePermissionAttribute class.

Declaration

// C#
public OraclePermissionAttribute (SecurityAction action);

Parameters

- **action**
  
  A System.Security.Permissions.SecurityAction value representing an action that can be performed using declarative security.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermissionAttribute Class
- OraclePermissionAttribute Members
OraclePermissionAttribute Static Methods

The OraclePermissionAttribute static methods are listed in Table 5–98.

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCustomAttribute</td>
<td>Inherited from System.Attribute (Overloaded)</td>
</tr>
<tr>
<td>GetCustomAttributes</td>
<td>Inherited from System.Attribute(Overloaded)</td>
</tr>
<tr>
<td>IsDefined</td>
<td>Inherited from System.Attribute(Overloaded)</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermissionAttribute Class
- OraclePermissionAttribute Members
OraclePermissionAttribute Public Properties

The OraclePermissionAttribute public properties are listed in Table 5–99.

<table>
<thead>
<tr>
<th>Public Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Inherited from SecurityAttribute</td>
</tr>
<tr>
<td>AllowBlankPassword</td>
<td>Inherited from DBDataPermissionAttribute. OraclePermissionAttribute ignores this property. Any value set for this property, for an OraclePermissionAttribute object, is ignored.</td>
</tr>
<tr>
<td>ConnectionString</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>KeyRestrictionBehavior</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>KeyRestrictions</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>TypeId</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Unrestricted</td>
<td>Inherited from SecurityAttribute</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermissionAttribute Class
- OraclePermissionAttribute Members
OraclePermissionAttribute Public Methods

The OraclePermissionAttribute public methods are listed in Table 5–100.

<table>
<thead>
<tr>
<th>Public Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatePermission</td>
<td>Returns a new OraclePermissionAttribute object that is configured based on</td>
</tr>
<tr>
<td></td>
<td>the attributes set</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Match</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ShouldSerializeConnectionString</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>ShouldSerializeKeyRestrictions</td>
<td>Inherited from DBDataPermissionAttribute</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

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

CreatePermission

This method returns a new OraclePermissionAttribute object that is configured based on the attributes set.

Declaration

```csharp
// C#
public override IPermission CreatePermission();
```

Return Value

An OraclePermission object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OraclePermissionAttribute Class
- OraclePermissionAttribute Members
OracleRowUpdatedEventArgs Class

The OracleRowUpdatedEventArgs class provides event data for the OracleDataAdapter.RowUpdated event.

Class Inheritance
System.Object
   System.EventArgs
      System.RowUpdatedEventArgs
         System.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 RowUpdated event "Example" on page 5-137.

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleRowUpdatedEventArgs Members
- OracleRowUpdatedEventArgs Constructor
- OracleRowUpdatedEventArgs Static Methods
- OracleRowUpdatedEventArgs Properties
- OracleRowUpdatedEventArgs Public Methods
- OracleDataAdapter Class on page 5-116
OracleRowUpdatedEventArgs Members

OracleRowUpdatedEventArgs members are listed in the following tables.

**OracleRowUpdatedEventArgs Constructors**

OracleRowUpdatedEventArgs constructors are listed in Table 5–101.

<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 method is listed in Table 5–102.

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

**OracleRowUpdatedEventArgs Properties**

The OracleRowUpdatedEventArgs properties are listed in Table 5–103.

<table>
<thead>
<tr>
<th>Property</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 System.Data.Common.RowUpdatedEventArgs</td>
</tr>
<tr>
<td>Row</td>
<td>Inherited from System.Data.Common.RowUpdatedEventArgs</td>
</tr>
<tr>
<td>StatementType</td>
<td>Inherited from System.Data.Common.RowUpdatedEventArgs</td>
</tr>
<tr>
<td>Status</td>
<td>Inherited from System.Data.Common.RowUpdatedEventArgs</td>
</tr>
<tr>
<td>TableMapping</td>
<td>Inherited from System.Data.Common.RowUpdatedEventArgs</td>
</tr>
</tbody>
</table>

**OracleRowUpdatedEventArgs Public Methods**

The OracleRowUpdatedEventArgs properties are listed in Table 5–104.

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleRowUpdatedEventArgs Class
OracleRowUpdatedEventArgs Constructor

The OracleRowUpdatedEventArgs constructor creates a new OracleRowUpdatedEventArgs instance.

Declaration

```csharp
// 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 1-4
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members
OracleRowUpdatedEventArgs Static Methods

The `OracleRowUpdatedEventArgs` static method is listed in Table 5-105.

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleRowUpdatedEventArgs Class`
- `OracleRowUpdatedEventArgs Members`
OracleRowUpdatedEventArgs Properties

The `OracleRowUpdatedEventArgs` properties are listed in Table 5–106.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong></td>
<td>Specifies the <code>OracleCommand</code> that is used when <code>OracleDataAdapter.Update()</code> is called</td>
</tr>
<tr>
<td><strong>Errors</strong></td>
<td>Inherited from <code>System.Data.Common.RowUpdatedEventArgs</code></td>
</tr>
<tr>
<td><strong>Row</strong></td>
<td>Inherited from <code>System.Data.Common.RowUpdatedEventArgs</code></td>
</tr>
<tr>
<td><strong>StatementType</strong></td>
<td>Inherited from <code>System.Data.Common.RowUpdatedEventArgs</code></td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Inherited from <code>System.Data.Common.RowUpdatedEventArgs</code></td>
</tr>
<tr>
<td><strong>TableMapping</strong></td>
<td>Inherited from <code>System.Data.Common.RowUpdatedEventArgs</code></td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `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 1-4
- `OracleRowUpdatedEventArgs Class`
- `OracleRowUpdatedEventArgs Members`
OracleRowUpdatedEventArgs Public Methods

The OracleRowUpdatedEventArgs properties are listed in Table 5–107.

Table 5–107  OracleRowUpdatedEventArgs Public Methods

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleRowUpdatedEventArgs Class
- OracleRowUpdatedEventArgs Members
The `OracleRowUpdatedEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdated` event.

**Declaration**

```csharp
// 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.

**Requirements**

- Assembly: `Oracle.DataAccess.dll`
- ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "RowUpdated" on page 5-136
OracleRowUpdatingEventArgs Class

The OracleRowUpdatingEventArgs class provides event data for the OracleDataAdapter.RowUpdating event.

Class Inheritance
System.Object
  System.EventArgs
    System.RowUpdatingEventArgs
      System.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 RowUpdated event "Example" on page 5-137.

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
  "Oracle.DataAccess.Client Namespace" on page 1-4
  OracleRowUpdatingEventArgs Members
  OracleRowUpdatingEventArgs Constructor
  OracleRowUpdatingEventArgs Static Methods
  OracleRowUpdatingEventArgs Properties
  OracleRowUpdatingEventArgs Public Methods
  "OracleDataAdapter Class" on page 5-116
OracleRowUpdatingEventArgs Members

OracleRowUpdatingEventArgs members are listed in the following tables.

**OracleRowUpdatingEventArgs Constructors**
OracleRowUpdatingEventArgs constructors are listed in Table 5–108.

**Table 5–108 OracleRowUpdatingEventArgs Constructors**

<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 5–109.

**Table 5–109 OracleRowUpdatingEventArgs Static Methods**

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

**OracleRowUpdatingEventArgs Properties**
The OracleRowUpdatingEventArgs properties are listed in Table 5–110.

**Table 5–110 OracleRowUpdatingEventArgs Properties**

<table>
<thead>
<tr>
<th>Property</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>Row</td>
<td>Inherited from System.Data.Common.RowUpdatingEventArgs</td>
</tr>
<tr>
<td>StatementType</td>
<td>Inherited from System.Data.Common.RowUpdatingEventArgs</td>
</tr>
<tr>
<td>Status</td>
<td>Inherited from System.Data.Common.RowUpdatingEventArgs</td>
</tr>
<tr>
<td>TableMapping</td>
<td>Inherited from System.Data.Common.RowUpdatingEventArgs</td>
</tr>
</tbody>
</table>

**OracleRowUpdatingEventArgs Public Methods**
The OracleRowUpdatingEventArgs public methods are listed in Table 5–111.

**Table 5–111 OracleRowUpdatingEventArgs Public Methods**

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>
OracleRowUpdatingEventArgs Class

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
- `OracleRowUpdatingEventArgs Class`
- `OracleRowUpdatingEventArgs Members`
OracleRowUpdatingEventArgs Static Methods

The OracleRowUpdatingEventArgs static method is listed in Table 5–112.

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleRowUpdatingEventArgs Class
- OracleRowUpdatingEventArgs Members
OracleRowUpdatingEventArgs Properties

The OracleRowUpdatingEventArgs properties are listed in Table 5–113.

<table>
<thead>
<tr>
<th>Property</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>Row</td>
<td>Inherited from System.Data.Common.RowUpdatingEventArgs</td>
</tr>
<tr>
<td>StatementType</td>
<td>Inherited from System.Data.Common.RowUpdatingEventArgs</td>
</tr>
<tr>
<td>Status</td>
<td>Inherited from System.Data.Common.RowUpdatingEventArgs</td>
</tr>
<tr>
<td>TableMapping</td>
<td>Inherited from System.Data.Common.RowUpdatingEventArgs</td>
</tr>
</tbody>
</table>

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

Command

This property specifies the OracleCommand that is used when the OracleDataAdapter.Update() is called.

Declaration

// C#
public new SqlCommand Command {get; set;}

Property Value
The OracleCommand executed when Update is called.

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

The OracleRowUpdatingEventArgs public methods are listed in Table 5–114.

Table 5–114  OracleRowUpdatingEventArgs Public Methods

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

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

The `OracleRowUpdatingEventHandler` delegate represents the signature of the method that handles the `OracleDataAdapter.RowUpdating` event.

**Declaration**

```csharp
// C#
public delegate void OracleRowUpdatingEventHandler (object sender,
                                         OracleRowUpdatingEventArgs eventArgs);
```

**Parameters**

- **sender**
  
  The source of the event.

- **eventArgs**
  
  The `OracleRowUpdatingEventArgs` object that contains the event data.

**Remarks**

Event callbacks can be registered through this event delegate for applications that wish to be notified after a row is updated.

In the .NET framework, the convention of an event delegate requires two parameters: the object that raises the event and the event data.

**Requirements**

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**

- "`Oracle.DataAccess.Client Namespace" on page 1-4"
- "`RowUpdating" on page 5-138"
OracleTransaction Class

An OracleTransaction object represents a local transaction.

Class Inheritance
System.Object
  System.MarshalByRefObject
    System.Data.Common.DbTransaction (ADO.NET 2.0 only)

Declaration
// ADO.NET 2.0: C#
public sealed class OracleTransaction : DbTransaction

// 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 Read Committed mode only. 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, if you have not done so yet.
/*
 connect scott/tiger@oracle
 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()
{
    // Drop & Create MyTable as indicated Database Setup, at beginning

    // 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();
        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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

Comment: Not supported in a .NET stored procedure

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleTransaction Members
- OracleTransaction Static Methods
- OracleTransaction Properties
OracleTransaction Members

OracleTransaction members are listed in the following tables.

OracleTransaction Static Methods
The OracleTransaction static method is listed in Table 5–115.

Table 5–115  OracleTransaction Static Method

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

OracleTransaction Properties
OracleTransaction properties are listed in Table 5–116.

Table 5–116  OracleTransaction Properties

<table>
<thead>
<tr>
<th>Property</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 5–117.

Table 5–117  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 System.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 System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.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 System.Object</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleTransaction Class
OracleTransaction Static Methods

The `OracleTransaction` static method is listed in Table 5-118.

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

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

OracleTransaction properties are listed in Table 5–119.

<table>
<thead>
<tr>
<th>Property</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 1-4
- OracleTransaction Class
- OracleTransaction Members

IsolationLevel

This property specifies the isolation level for the transaction.

Declaration

// ADO.NET 2.0: C#
public override 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 1-4
- 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 1-4
- OracleTransaction Class
- OracleTransaction Members
OracleTransaction public methods are listed in Table 5–120.

### Table 5–120   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 System.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 System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.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 System.Object</td>
</tr>
</tbody>
</table>

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

### Commit

This method commits the database transaction.

**Declaration**

```csharp
// ADO.NET 2.0: C#
public override 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**

```csharp
// Database Setup, if you have not done so yet
/*
connect scott/tiger@oracle
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 in Database Setup, at beginning

        // 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();
            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 1-4
- 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 1-4
- 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
// ADO.NET 2.0: C#
public override 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
// Database Setup, if you have not done so yet.
/*
connect scott/tiger@oracle
DROP TABLE MyTable;
CREATE TABLE MyTable (MyColumn NUMBER);
*/

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

class RollbackSample
{
    static void Main()
    {
        // Drop & Create MyTable as indicated previously in Database Setup

        // This sample starts a transaction and inserts one record into MyTable.
        // It then rollback 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

        See Also:
        ■ "Oracle.DataAccess.Client Namespace" on page 1-4
        ■ OracleTransaction Class
        ■ OracleTransaction Members
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();

// 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 1-4
- OracleTransaction Class
- OracleTransaction Members

Rollback(string)

This method rolls back a database transaction to a savepoint within the current transaction.

Declaration

// ADO.NET 2.0: C#
public override 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.
**OracleTransaction Class**

**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 1-4
- OracleTransaction Class
- OracleTransaction Members

**Save**

This method creates a `savepoint` within the current transaction.

**Declaration**

```csharp
// 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**

```csharp
// Database Setup, if you have not done so yet.
/*
connect scott/tiger@oracle
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 in Database Setup, at beginning

    // This sample starts a transaction and creates a savepoint after
    // inserting one record into MyTable.
    // After inserting the second record it rollsback 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 1-4
- OracleTransaction Class
- OracleTransaction Members
OracleConnectionType Enumeration

OracleConnectionType enumeration values specify whether a particular connection object is associated with an Oracle database connection, a TimesTen database connection, or no physical connection at all.

Table 5–121 lists all the OracleConnectionType enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undefined</td>
<td>No connection is associated with the OracleConnection object.</td>
</tr>
<tr>
<td>Oracle</td>
<td>The OracleConnection object is associated with an Oracle database.</td>
</tr>
<tr>
<td>TimesTen</td>
<td>The OracleConnection object is associated with a TimesTen database.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleConnection Class" on page 5-64
- "ConnectionType" on page 5-84
OracleCollectionType Enumeration

OracleCollectionType enumerated values specify whether or not the OracleParameter object represents a collection, and if so, specifies the collection type.

Table 5–122 lists all the OracleCollectionType enumeration values with a description of each enumerated value.

Table 5–122  OracleCollectionType Enumeration Values

<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>

Requirements

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleParameter Class" on page 5-248
- "CollectionType" on page 5-267
OracleDBShutdownMode Enumeration

OracleDBShutdownMode enumerated values specify the database shutdown options.

Table 5–124 lists all the OracleDBShutdownMode enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Refuses new connections and waits for existing connections to end.</td>
</tr>
<tr>
<td>Transactional</td>
<td>Refuses new connections and does not allow any new transactions. Waits for active transactions to commit.</td>
</tr>
<tr>
<td>TransactionalLocal</td>
<td>Refuses new connections and does not allow any new transactions. Waits for only local transactions to commit.</td>
</tr>
<tr>
<td>Immediate</td>
<td>Does not wait for current calls to complete or users to disconnect from the database. All uncommitted transactions are terminated and rolled back.</td>
</tr>
<tr>
<td>Final</td>
<td>Shuts down the database. Used in the second call for shutdown after the database has been closed and dismounted.</td>
</tr>
<tr>
<td>Abort</td>
<td>Does not wait for current calls to complete or users to disconnect from the database. All uncommitted transactions are terminated and are not rolled back.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleConnection Class" on page 5-64
- "Shutdown" on page 5-145
OracleDBStartupMode Enumeration

OracleDBStartupMode enumerated values specify the database startup options. Table 5–124 lists all the OracleDBStartupMode enumeration values with a description of each enumerated value.

Table 5–124 OracleDBStartupMode Enumeration Values

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoRestriction</td>
<td>Starts the database and allows access to all users.</td>
</tr>
<tr>
<td>Restrict</td>
<td>Starts the database and allows database access only to users having the CREATE SESSION and RESTRICTED SESSION privileges. These privileges are normally assigned to database administrators.</td>
</tr>
<tr>
<td>Force</td>
<td>Shuts down a running instance in abort mode and starts a new instance.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleConnection Class" on page 5-64
- "Startup" on page 5-149
OracleDbType Enumeration

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

Table 5–125 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>Array</td>
<td>Oracle Collection (VArray or Nested Table)</td>
</tr>
<tr>
<td>BFile</td>
<td>Oracle BFILE type</td>
</tr>
<tr>
<td>BinaryFloat</td>
<td>Oracle BINARY_FLOAT type</td>
</tr>
<tr>
<td>BinaryDouble</td>
<td>Oracle BINARY_DOUBLE 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>Object</td>
<td>Oracle Object</td>
</tr>
<tr>
<td>Raw</td>
<td>Oracle RAW type</td>
</tr>
<tr>
<td>Ref</td>
<td>Oracle REF</td>
</tr>
<tr>
<td>RefCursor</td>
<td>Oracle REF CURSOR type</td>
</tr>
<tr>
<td>Single</td>
<td>4-byte FLOAT type, supports 6 precisions</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>
</tbody>
</table>
Table 5–125 (Cont.) OracleDbType Enumeration Values

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XmlType</td>
<td>Oracle XMLType type</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace” on page 1-4
- "OracleParameter Class” on page 5-248
- "OracleParameterCollection Class” on page 5-283
- OracleParameter "OracleDbType" on page 5-270
The `OracleParameterStatus` enumeration type indicates whether a NULL value is fetched from a column, or truncation has occurred during the fetch, or a NULL value is to be inserted into a database column. Table 5–126 lists all the `OracleParameterStatus` enumeration values with a description of each enumerated value.

### Table 5–126  `OracleParameterStatus` Members

<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>

### Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "`Oracle.DataAccess.Client Namespace"` on page 1-4
- "`OracleParameter Class"" on page 5-248
- `OracleParameter "ArrayBindStatus"` on page 5-266
- `OracleParameter "Value"` on page 5-277
Oracle Data Provider for .NET XML-Related Classes

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 6–1 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 1-4
OracleXmlQueryProperties Class

An OracleXmlQueryProperties object represents the XML properties used by the OracleCommand class when the XmlCommandType property is Query.

Class Inheritance
System.Object
  System.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 System.IO;
using System.Data;
using System.Xml;
using System.Text;
using Oracle.DataAccess.Client;

class OracleXmlQueryPropertiesSample
{
  static void Main()
  {
    int rows = 0;
    StreamReader sr = null;

    // Define the XSL document for doing the transform.
    string xslstr = "<?xml version='1.0'?>\n" +
      "<xsl:stylesheet version='1.0'" +
      " xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>\n" +
      " <xsl:output encoding='utf-8'/>\n" +
      " <xsl:template match='\/*'>\n" +
      " <EMPLOYEES>\n" +
      "   <xsl:apply-templates select='\"ROWSET\"'/>\n" +
      " </EMPLOYEES>\n" +
      "</xsl:stylesheet>\n" +
"}
" </xsl:template>
" </xsl:template>
" <xsl:template match="ROWSET">
"      <xsl:apply-templates select="ROW"/>
"  </xsl:template>
"  <xsl:template match="ROW">
"    <EMPLOYEE>
"    <EMPLOYEE_ID>
"      <xsl:apply-templates select="EMPNO"/>
"    </EMPLOYEE_ID>
"    <EMPLOYEE_NAME>
"      <xsl:apply-templates select="ENAME"/>
"    </EMPLOYEE_NAME>
"    <HIRE_DATE>
"      <xsl:apply-templates select="HIREDATE"/>
"    </HIRE_DATE>
"    <JOB_TITLE>
"      <xsl:apply-templates select="JOB"/>
"    </JOB_TITLE>
"    </EMPLOYEE>
"  </xsl:template>
"</xsl:stylesheet>

// Create the connection.
string constr = "User Id=scott;Password=tiger;Data Source=oracle";
OracleConnection con = new OracleConnection{constr};
con.Open();

// Set the date, and timestamp formats for Oracle 9i Release 2, or later.
// This is just needed for queries.
if (!con.ServerVersion.StartsWith("9.0") &&
    !con.ServerVersion.StartsWith("8.1"))
{
    OracleGlobalization sessionParams = con.GetSessionInfo();
    sessionParams.DateFormat = "YYYY-MM-DD"T"HH24:MI:SS";
    sessionParams.TimeStampFormat = "YYYY-MM-DD"T"HH24:MI:SS.FF3";
    sessionParams.TimeStampTZFormat = "YYYY-MM-DD"T"HH24:MI:SS.FF3";
    con.SetSessionInfo(sessionParams);
}

// 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 emp e where e.empno = :empno";

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

// Bind values to the parameters in the SQL query.
Int32 empNum = 7369;
cmd.Parameters.Add("empno", OracleDbType.Int32, empNum, ParameterDirection.Input);

// Set the XML query properties.
cmd.XmlQueryProperties.MaxRows = 1;
cmd.XmlQueryProperties.RootTag = "ROWSET";
cmd.XmlQueryProperties.RowTag = "ROW";
cmd.XmlQueryProperties.Xslt = xslstr;

// Test query execution without returning a result.
Console.WriteLine("SQL query: select * from emp e where e.empno = 7369");
Console.WriteLine("Maximum rows: all rows (-1)");
Console.WriteLine("Return Value from OracleCommand.ExecuteNonQuery()");
rows = cmd.ExecuteNonQuery();
Console.WriteLine(rows);
Console.WriteLine("\n");

// Get the XML document as an XmlReader.
Console.WriteLine("SQL query: select * from emp e where e.empno = 7369");
Console.WriteLine("Maximum rows: all rows (-1)");
Console.WriteLine("XML Document from OracleCommand.ExecuteXmlReader()");

XmlReader xmlReader = cmd.ExecuteXmlReader();
XmlDocument xmlDocument = new XmlDocument();
xmlDocument.PreserveWhitespace = true;
xmlDocument.Load(xmlReader);
Console.WriteLine(xmlDocument.OuterXml);
Console.WriteLine("\n");

// Change the SQL query, and set the maximum number of rows to 2.
   cmd.CommandText = "select * from emp e";
   cmd.Parameters.Clear();
   cmd.XmlQueryProperties.MaxRows = 2;

// Get the XML document as a Stream.
Console.WriteLine("SQL query: select * from emp e");
Console.WriteLine("Maximum rows: 2");
Stream stream = cmd.ExecuteStream();
sr = new StreamReader(stream, Encoding.Unicode);
Console.WriteLine(sr.ReadToEnd());
Console.WriteLine("\n");

// Get all the rows.
cmd.XmlQueryProperties.MaxRows = -1;

// Append the XML document to an existing Stream.
Console.WriteLine("SQL query: select * from emp e");
Console.WriteLine("Maximum rows: all rows (-1)");
MemoryStream mstream = new MemoryStream(32);
cmd.ExecuteToStream(mstream);
mstream.Seek(0, SeekOrigin.Begin);
sr = new StreamReader(mstream, Encoding.Unicode);
Console.WriteLine(sr.ReadToEnd());
Console.WriteLine("\n");

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

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 6–2.

Table 6–2 OracleXmlQueryProperties Constructors

<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 6–3.

Table 6–3 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>

OracleXmlQueryProperties Public Methods
The OracleXmlQueryProperties public methods are listed in Table 6–4.

Table 6–4 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>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleXmlQueryProperties Class
OracleXmlQueryProperties Constructor

The `OracleXmlQueryProperties` constructor instantiates a new instance of the `OracleXmlQueryProperties` class.

Declaration

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
OracleXmlQueryProperties Class

OracleXmlQueryProperties Properties

The OracleXmlQueryProperties properties are listed in Table 6–5.

<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 1-4
- 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

```csharp
// 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 1-4
- 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 1-4
- 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 1-4
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members

Xslt

This property specifies the XSL document used for XML transformation using XSLT.
OracleXmlQueryProperties Class

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 1-4
- 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 1-4
- OracleXmlQueryProperties Class
- OracleXmlQueryProperties Members
OracleXmlQueryProperties Public Methods

The OracleXmlQueryProperties public methods are listed in Table 6–6.

<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 1-4
- 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
System.Object

    System.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#
    /* -- If HR account is being locked, you need to log on as a DBA
       -- to unlock the account first. Unlock a locked users account:

       ALTER USER hr ACCOUNT UNLOCK;
    */

    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"?>\n" +
"<ROWSET>\n" +
" <MYROW num = "1">\n" +
" <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
" <LAST_NAME>Smith</LAST_NAME>\n" +
" <EMAIL>Smith@Oracle.com</EMAIL>\n" +
" <HIRE_DATE>1982-01-23T00:00:00.000</HIRE_DATE>\n" +
" <JOB_ID>IT_PROG</JOB_ID>\n" +
" </MYROW>\n" +
" <MYROW num = "2">\n" +
" <EMPLOYEE_ID>1235</EMPLOYEE_ID>\n" +
" <LAST_NAME>Barney</LAST_NAME>\n" +
" <EMAIL>Barney@Oracle.com</EMAIL>\n" +
" <HIRE_DATE>1982-01-23T00:00:00.000</HIRE_DATE>\n" +
" <JOB_ID>IT_PROG</JOB_ID>\n" +
" </MYROW>\n" +
"</ROWSET>\n";

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";
UpdateColumnsList = new string[5];
UpdateColumnsList[0] = "EMPLOYEE_ID";
UpdateColumnsList[1] = "LAST_NAME";
UpdateColumnsList[2] = "EMAIL";
UpdateColumnsList[3] = "HIRE_DATE";
UpdateColumnsList[4] = "JOB_ID";
cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
cmd.XmlSaveProperties.RowTag = "MYROW";
cmd.XmlSaveProperties.Table = "employees";
cmd.XmlSaveProperties.UpdateColumnsList = UpdateColumnsList;
cmd.XmlSaveProperties.Xslt = null;
cmd.XmlSaveProperties.XsltParams = null;

// Do the inserts.
rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Set the XML command type to update.
cmd.XmlCommandType = OracleXmlCommandType.Update;

// Set the XML document.
cmd.CommandText = "<?xml version="1.0"?>\n" +
"<ROWSET>\n" +
" <MYROW num = "1">\n" +
" <EMPLOYEE_ID>1234</EMPLOYEE_ID>\n" +
" <LAST_NAME>Adams</LAST_NAME>\n" +
" </MYROW>\n" +
"</ROWSET>\n";

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";
UpdateColumnsList = new string[1];
UpdateColumnsList[0] = "LAST_NAME";

cmd.XmlSaveProperties.KeyColumnsList = KeyColumnsList;
cmd.XmlSaveProperties.UpdateColumnsList = UpdateColumnsList;
rows = cmd.ExecuteNonQuery();
Console.WriteLine("rows: " + rows);

// Set the XML command type to delete.
cmd.XmlCommandType = OracleXmlCommandType.Delete;

// Set the XML document.

// Set the XML save properties.
KeyColumnsList = new string[1];
KeyColumnsList[0] = "EMPLOYEE_ID";

// 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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 6–7.

Table 6–7 OracleXmlSaveProperties Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleXmlSaveProperties</td>
<td>Instantiates a new instance of the OracleXmlSaveProperties class</td>
</tr>
<tr>
<td>Constructor</td>
<td></td>
</tr>
</tbody>
</table>

OracleXmlSaveProperties Properties

The OracleXmlSaveProperties properties are listed in Table 6–8.

Table 6–8 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 6–9.

Table 6–9 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 1-4
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members
OracleXmlSaveProperties Constructor

The OracleXmlSaveProperties constructor instantiates a new instance of the OracleXmlSaveProperties class.

Declaration

\[
\text{C#}\]
public OracleXmlSaveProperties;

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members
OracleXmlSaveProperties Properties

The OracleXmlSaveProperties properties are listed in Table 6–10.

Table 6–10 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</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 1-4
- 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

```csharp
// 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.
RowTag

This property specifies the value for the XML element that identifies a row of data in an XML document.

Declaration

// C#
public string RowTag {get; set;}

Property Value

An XML element name.

Remarks

The default value is "ROW".

Each element in the XML document identifies one row in a table or view.

If RowTag is set to "" or null, no row tag is used in the XML document. In this case, the XML document is assumed to contain only one row.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members

Table

This property specifies the name of the table or view to which changes are saved.

Declaration

// C#
public string Table {get; set;}

Property Value

A table name.

Remarks

Default value is null.

The property must be set to a valid table or view name.

See Also:

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

UpdateColumnsList usage with XMLCommandType property values:

- **Insert** - UpdateColumnsList indicates which columns are assigned values when a new row is created. If UpdateColumnsList is null, then all columns are assigned values. If a column is on the UpdateColumnsList, but no value is specified for the row in the XML file, then NULL is used. If a column is not on the UpdateColumnsList, then the default value for that column is used.

- **Update** - UpdateColumnsList specifies columns to modify for each row of data in the XML document. If UpdateColumnsList 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 1-4
- 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.
**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 1-4
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members
OracleXmlSaveProperties Public Methods

The OracleXmlSaveProperties public methods are listed in Table 6–11.

Table 6–11  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>

Clone

This method creates a copy of an OracleXmlSaveProperties object.

Declaration

// C#
public object Clone();

Return Value

An OracleXmlSaveProperties object

Implements

ICloneable

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleXmlSaveProperties Class
- OracleXmlSaveProperties Members
OracleXmlStream Class

An OracleXmlStream object represents a read-only stream of XML data stored in an OracleXmlType object.

Class Inheritance
System.Object
   System.MarshalByRefObject
   System.Stream
   System.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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleXmlStream Members
- OracleXmlStream Constructor
- OracleXmlStream Static Methods
- OracleXmlStream Instance Properties
- OracleXmlStream Instance Methods
OracleXmlStream Members

OracleXmlStream members are listed in the following tables.

OracleXmlStream Constructors
The OracleXmlStream constructors are listed in Table 6–12.

Table 6–12 OracleXmlStream Constructors

<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 6–13.

Table 6–13 OracleXmlStream Static Methods

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

OracleXmlStream Instance Properties
The OracleXmlStream instance properties are listed in Table 6–14.

Table 6–14 OracleXmlStream Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether or not the XML stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether or not 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 6–15.

Table 6–15 OracleXmlStream Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from System.IO.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>
### OracleXmlStream Class

**Table 6–15  (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 System.IO.Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Flush</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.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 System.IO.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 System.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 1-9
- 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 1-9
- OracleXmlStream Class
- OracleXmlStream Members
OracleXmlStream Class

OracleXmlStream Static Methods

The OracleXmlStream static methods are listed in Table 6–16.

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleXmlStream Class
- OracleXmlStream Members
OracleXmlStream Instance Properties

The OracleXmlStream instance properties are listed in Table 6–17.

Table 6–17  OracleXmlStream Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether or not the XML stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether or not 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 1-9
- OracleXmlStream Class
- OracleXmlStream Members

CanRead

Overrides Stream

This property indicates whether or not the XML stream can be read.

Declaration

```csharp
// C#
public override bool CanRead();
```

Property Value

If the XML stream is can be read, returns true; otherwise, returns false.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleXmlStream Class
- OracleXmlStream Members

CanSeek

Overrides Stream

This property indicates whether or not 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 1-9
- 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 1-9
- OracleXmlStream Class
- OracleXmlStream Members

**Length**
Overrides Stream
This property indicates the number of bytes in the XML stream.

**Declaration**
```
// C#
public override Int64 Length {get;}
```

**Property Value**
An Int64 value representing the number of bytes in the XML stream. An empty stream has a length of 0 bytes.

**Exceptions**
ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleXmlStream Class
- OracleXmlStream Members

Position

Overrides Stream
This property gets or sets the byte position within the stream.

Declaration
// C#
public override Int64 Position{get; set;}

Property Value
An Int64 that indicates the current position in the stream.

Exceptions
ObjectDisposedException - The object is already disposed.
InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.
ArgumentOutOfRangeException - The Position is less than 0.

Remarks
The beginning of the stream is represented by position 0. Seeking to any location beyond the length of the stream is supported.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleXmlStream Class
- OracleXmlStream Members

Value

This property returns the XML data, starting from the first character of the stream as a string.

Declaration
// C#
public string Value{get; set;}

Property Value
A string.

Exceptions
ObjectDisposedException - The object is already disposed.
InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.
Remarks
The value of Position is neither used nor changed by using this property.
The maximum length of the string that can be returned by this property is 2 GB.

See Also:
- "Oracle.DataAccess.Types Namespace” on page 1-9
- OracleXmlStream Class
- OracleXmlStream Members
OracleXmlStream Instance Methods

The OracleXmlStream instance methods are listed in Table 6–18.

Table 6–18 OracleXmlStream Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from System.IO.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</td>
</tr>
<tr>
<td></td>
<td>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 System.IO.Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Flush</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Read</td>
<td>Reads a specified amount from the current XML stream</td>
</tr>
<tr>
<td></td>
<td>instance and populates the array buffer (Overloaded)</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>Seek</td>
<td>Sets the position within the current stream and returns the</td>
</tr>
<tr>
<td></td>
<td>new position within the current stream</td>
</tr>
<tr>
<td>SetLength</td>
<td>Not Supported</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.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 1-9
- OracleXmlStream Class
- OracleXmlStream Members

Clone

This method creates a copy of an OracleXmlStream object.

Declaration

```csharp
// C#
public object Clone();
```
OracleXmlStream Class

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 1-9
- 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 1-9
- 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 Instance Methods

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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

```csharp
// 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 1-9
- 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 1-9
- `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 or not 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 1-9
- `OracleXmlStream` Class
- `OracleXmlStream` Members
OracleXmlType Class

An OracleXmlType object represents an Oracle XMLType instance.

Class Inheritance
System.Object
    System.OracleXmlType

Declaration
// C#
public sealed class OracleXmlType : IDisposable, INullable

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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 6–19.

Table 6–19 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 6–20.

Table 6–20 OracleXmlType Static Methods

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

OracleXmlType Instance Properties
The OracleXmlType instance properties are listed in Table 6–21.

Table 6–21 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 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>

OracleXmlType Instance Methods
The OracleXmlType instance methods are listed in Table 6–22.

Table 6–22 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 6–22 (Cont.) OracleXmlType Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.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 System.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 System.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 System.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 or not 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 1-9
- OracleXmlType Class
OracleXmlType Constructors

OracleXmlType constructors create instances of the OracleXmlType class.

Overload List:

- **OracleXmlType(OracleClob)**
  This constructor creates an instance of the OracleXmlType class using the XML data contained in an OracleClob object.

- **OracleXmlType(OracleConnection, string)**
  This constructor creates an instance of the OracleXmlType class using the XML data contained in the .NET String.

- **OracleXmlType(OracleConnection, XmlReader)**
  This constructor creates an instance of the OracleXmlType class using the contents of the .NET XmlReader object.

- **OracleXmlType(OracleConnection, XmlDocument)**
  This constructor creates an instance of the OracleXmlType object using the contents of the XML DOM document in the .NET XmlDocument object.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleXmlType Class
- OracleXmlType Members

OracleXmlType(OracleClob)

This constructor creates an instance of the OracleXmlType class using the XML data contained in an OracleClob object.

Declaration

```csharp
// C#
public OracleXmlType(OracleClob oraClob);
```

Parameters

- **oraClob**
  An OracleClob object.

Exceptions

- **ArgumentNullException** - The OracleClob object is null.
- **InvalidOperationException** - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The CLOB data depends on a valid connection object and the new OracleXMLType uses the OracleConnection in the OracleClob object to store data for the current instance.
OracleXmlType Class

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 1-9
- 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(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 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 1-9

■ OracleXmlType Class

■ OracleXmlType Members
OracleXmlType Static Methods

The OracleXmlType static methods are listed in Table 6-23.

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleXmlType Class
- OracleXmlType Members
OracleXmlType Instance Properties

The OracleXmlType instance properties are listed in Table 6–24.

Table 6–24  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 1-9
- OracleXmlType Class
- OracleXmlType Members

Connection

This property indicates the OracleConnection that is used to retrieve and store XML data in the OracleXmlType.

Declaration

// C#
public OracleConnection Connection {get;}

Property Value
An OracleConnection object.

Exceptions
ObjectDisposedException - The object is already disposed.

Remarks
The connection must explicitly be opened by the user before creating or using OracleXmlType.
IsEmpty

This property indicates whether or not the OracleXmlType is empty.

**Declaration**

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

**Property Value**

Returns `true` if the OracleXmlType represents an empty XML document. Returns `false` otherwise.

**Exceptions**

- **ObjectDisposedException** - The object is already disposed.
- **InvalidOperationException** - The OracleConnection is not open or has been closed during the lifetime of the object.

See Also:

- [Oracle.DataAccess.Types Namespace](#) on page 1-9
- OracleXmlType Class
- OracleXmlType Members

IsFragment

This property indicates whether the XML data is a collection of XML elements or a well-formed XML document.

**Declaration**

```csharp
// 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 1-9
- OracleXmlType Class
- OracleXmlType Members
OracleXmlType Instance Properties

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 1-9
- 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 1-9
- OracleXmlType Class
- OracleXmlType Members

Schema
This property represents the XML schema for the XML data contained in the OracleXmlType.
Declaration

// C#
public OracleXmlType Schema {get;}

Property Value
An OracleXmlType instance that represents the XML schema for the XML data contained in the OracleXmlType.

Exceptions
ObjectDisposedException - The object is already disposed.

Remarks
If the OracleXmlType instance contains non-schema based XML data, this property returns an OracleXmlType instance representing an empty XML document.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleXmlType Class
- OracleXmlType Members
OracleXmlType Instance Methods

The OracleXmlType instance methods are listed in Table 6–25.

<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 System.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 System.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 System.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 XMLdata (Overloaded)</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.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 or not 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 1-9
- OracleXmlType Class
- OracleXmlType Members

Clone

This method creates a copy of this OracleXmlType instance.

Declaration

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

Implements

ICloneable
**OracleXmlType Instance Methods**

**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 1-9
- OracleXmlType Class
- OracleXmlType Members

**Dispose**
This method releases the resources allocated by this object.

**Declaration**

```csharp
// C#
public void Dispose();
```

**Implements**
IDisposable

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleXmlType Class
- OracleXmlType Members

Extract(string, XmlNameSpaceManager)

This public method extracts a subset from the XML data represented by the OracleXmlType object, using the given XPath expression and a .NET XmlNameSpaceManager object for namespace resolution.

Declaration

// C#
public OracleXmlType Extract(string xpathExpr, XmlNameSpaceManager nsMgr);

Parameters

- **xpathExpr**
  The XPath expression.

- **nsMgr**
  The .NET XmlNameSpaceManager object used for namespace resolution.
The .NET XmlNameSpaceManager object used for namespace resolution of the XPath expression. \textit{nsMgr} can be null.

Return Value
An OracleXmlType.

Exceptions
ObjectDisposedException - The object is already disposed.
ArgumentException - The \textit{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 1-9
- 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

\begin{verbatim}
// C#
public Stream GetStream();
\end{verbatim}

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 1-9
- OracleXmlType Class
- OracleXmlType Members

GetXmlDocument
This public method returns a XmlDocument object containing the XML data stored in this OracleXmlType instance.

Declaration

\begin{verbatim}
// C#
public XmlDocument GetXmlDocument();
\end{verbatim}
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 1-9
- 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 1-9
- 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.
OracleXmlType Instance Methods

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 1-9
- 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.
- 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 1-9
- 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 1-9
  - 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.
- InvalidOperation - The OracleConnection is not open or has been closed during the lifetime of the object.

  **See Also:**
  - "Oracle.DataAccess.Types Namespace" on page 1-9
  - OracleXmlType Class
  - OracleXmlType Members
Transform(string, string)

This method transforms the current `OracleXmlType` instance into another `OracleXmlType` instance using the given XSL document and a string of XSLT parameters.

**Declaration**

// C#
public OracleXmlType Transform(string xsldoc, string paramMap);

**Parameters**

- `xsldoc`
  - The XSL document to be used for XSLT.

- `paramMap`
  - A string which provides the parameters for the XSL document.
  - For this release, `paramMap` is ignored.

**Return Value**

An `OracleXmlType` object containing the transformed XML document.

**Exceptions**

ObjectDisposedException - The object is already disposed.

ArgumentNullException - The `xsldoc` parameter is null.

InvalidOperationException - The `OracleConnection` is not open or has been closed during the lifetime of the object.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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.
OracleXmlType Instance Methods

- **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 1-9
  - 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:
  

- **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 1-9
- 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 1-9
- 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
  

Oracle Data Provider for .NET XML-Related Classes 6-59
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.

\begin{itemize}
  \item \textbf{value}
  
  The new value as an \texttt{OracleXmlType} object.
\end{itemize}

\textbf{Exceptions}

- \texttt{ObjectDisposedException} - The object is already disposed.
- \texttt{ArgumentNullException} - The \texttt{xpathExpr} is null or zero-length.
- \texttt{InvalidOperationException} - The \texttt{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:}

- "\texttt{Oracle.DataAccess.Types Namespace}" on page 1-9
- \texttt{OracleXmlType Class}
- \texttt{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 \texttt{OracleXmlType} value and a .NET \texttt{XmlNameSpaceManager} object for namespace resolution.

\textbf{Declaration}

\texttt{// C#}
\begin{verbatim}
public void Update(string xpathExpr, XmlNameSpaceManager nsMgr, OracleXmlType value);
\end{verbatim}

\textbf{Parameters}

\begin{itemize}
  \item \texttt{xpathExpr}
    
    The XPath expression that identifies the nodes to update.
  
  \item \texttt{nsMgr}
    
    The .NET \texttt{XmlNameSpaceManager} object used for namespace resolution of the XPath expression. \texttt{nsMgr} can be null.
  
  \item \texttt{value}
    
    The new value as an \texttt{OracleXmlType} object.
\end{itemize}

\textbf{Exceptions}

- \texttt{ObjectDisposedException} - The object is already disposed.
- \texttt{ArgumentNullException} - The \texttt{xpathExpr} is null or zero-length.
- \texttt{InvalidOperationException} - The \texttt{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 or not 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.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleXmlType Class
- OracleXmlType Members
This chapter describes the following Oracle Data Provider for .NET classes that support the ADO.NET 2.0 specification.

See Also:  "ADO.NET 2.0 Features" on page 3-20

- OracleClientFactory Class
- OracleConnectionStringBuilder Class
- OracleDataSourceEnumerator Class
OracleClientFactory Class

An OracleClientFactory object allows applications to instantiate ODP.NET classes in a generic way.

Supported Only in ADO.NET 2.0-Compliant ODP.NET

Class Inheritance
System.Object

Declaration

    // C#
    public sealed class OracleClientFactory : DbProviderFactory

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

    class FactorySample
    {
        static void Main()
        {
            string constr = "user id=scott;password=tiger;data source=oracle";

            DbProviderFactory factory =

            DbConnection conn = factory.CreateConnection();

            try
            {
                conn.ConnectionString = constr;
                conn.Open();

                DbCommand cmd = factory.CreateCommand();
                cmd.Connection = conn;
                cmd.CommandText = "select * from emp";

                DbDataReader reader = cmd.ExecuteReader();
                while (reader.Read())
                {
                    Console.WriteLine(reader["EMPNO"] + " : " + reader["ENAME"]);
                }
            }
            catch (Exception ex)
            {
            
        }
Console.WriteLine(ex.Message);
Console.WriteLine(ex.StackTrace);
}
}

**Requirements**

**Namespace:** Oracle.DataAccess.Client

**Assembly:** Oracle.DataAccess.dll

**ODP.NET Version:** ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Members
- OracleClientFactory Public Properties
- OracleClientFactory Public Methods
OracleClientFactory Members

OracleClientFactory members are listed in the following tables.

OracleClientFactory Public Properties
The OracleClientFactory public properties are listed in Table 7–1.

Table 7–1 OracleClientFactory Public Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCreateDataSourceEnumerator</td>
<td>Indicates whether or not the CreateDataSourceEnumerator method is supported</td>
</tr>
</tbody>
</table>

OracleClientFactory Public Methods
OracleClientFactory Public Methods are listed in Table 7–2.

Table 7–2 OracleClientFactory Public Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateCommand</td>
<td>Returns a DbCommand object that represents an OracleCommand object</td>
</tr>
<tr>
<td>CreateCommandBuilder</td>
<td>Returns a DbCommandBuilder object that represents an OracleCommandBuilder object</td>
</tr>
<tr>
<td>CreateConnection</td>
<td>Returns a DbConnection object that represents an OracleConnection object</td>
</tr>
<tr>
<td>CreateConnectionStringBuilder</td>
<td>Returns a DbConnectionStringBuilder object that represents an OracleConnectionStringBuilder object</td>
</tr>
<tr>
<td>CreateDataAdapter</td>
<td>Returns a DbDataAdapter object that represents an OracleDataAdapter object</td>
</tr>
<tr>
<td>CreateDataSourceEnumerator</td>
<td>Returns a DbDataSourceEnumerator object that represents an OracleDataSourceEnumerator object</td>
</tr>
<tr>
<td>CreateParameter</td>
<td>Returns a DbParameter object that represents an OracleParameter object</td>
</tr>
<tr>
<td>CreatePermission</td>
<td>Returns a CodeAccessPermission object that represents an OraclePermission object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Class
OracleClientFactory Public Properties

The OracleClientFactory public properties are listed in Table 7–3.

Table 7–3  OracleClientFactory Public Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCreateDataSourceEnumerator</td>
<td>Indicates whether or not the CreateDataSourceEnumerator method is supported</td>
</tr>
</tbody>
</table>

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

CanCreateDataSourceEnumerator

This property indicates whether or not the CreateDataSourceEnumerator method is supported.

Declaration

// C#
public override bool CanCreateDataSourceEnumerator { get; }

Property Value
Returns true.

Remarks

ODP.NET supports the OracleDataSourceEnumerator object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Class
- OracleClientFactory Members
OracleClientFactory Public Methods

The OracleClientFactory public method is listed in Table 7–4.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateCommand</td>
<td>Returns a DbCommand object that represents an OracleCommand object.</td>
</tr>
<tr>
<td>CreateCommandBuilder</td>
<td>Returns a DbCommandBuilder object that represents an OracleCommandBuilder object.</td>
</tr>
<tr>
<td>CreateConnection</td>
<td>Returns a DbConnection object that represents an OracleConnection object.</td>
</tr>
<tr>
<td>CreateConnectionStringBuilder</td>
<td>Returns a DbConnectionStringBuilder object that represents an OracleConnectionStringBuilder object.</td>
</tr>
<tr>
<td>CreateDataAdapter</td>
<td>Returns a DbDataAdapter object that represents an OracleDataAdapter object.</td>
</tr>
<tr>
<td>CreateDataSourceEnumerator</td>
<td>Returns a DbDataSourceEnumerator object that represents an OracleDataSourceEnumerator object.</td>
</tr>
<tr>
<td>CreateParameter</td>
<td>Returns a DbParameter object that represents an OracleParameter object.</td>
</tr>
<tr>
<td>CreatePermission</td>
<td>Returns a CodeAccessPermission object that represents an OraclePermission object.</td>
</tr>
</tbody>
</table>

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

CreateCommand

This method returns a DbCommand object that represents an OracleCommand object.

Declaration

```csharp
// C#
public override DbCommand CreateCommand();
```

Return Value

A DbCommand object that represents an OracleCommand object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Class
- OracleClientFactory Members
CreateCommandBuilder

This method returns a DbCommandBuilder object that represents an OracleCommandBuilder object.

Declaration

// C#
public override DbCommandBuilder CreateCommandBuilder();

Return Value

A DbCommandBuilder object that represents an OracleCommandBuilder object.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Class
- OracleClientFactory Members

CreateConnection

This method returns a DbConnection object that represents an OracleConnection object.

Declaration

// C#
public override DbConnection CreateConnection();

Return Value

A DbConnection object that represents an OracleConnection object.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Class
- OracleClientFactory Members

CreateConnectionStringBuilder

This method returns a DbConnectionStringBuilder object that represents an OracleConnectionStringBuilder object.

Declaration

// C#
public override DbConnectionStringBuilder CreateConnectionStringBuilder();

Return Value

A DbConnectionStringBuilder object that represents an OracleConnectionStringBuilder object.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Class
- OracleClientFactory Members
OracleClientFactory Public Methods

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

CreateDataAdapter
This method returns a DbDataAdapter object that represents an OracleDataAdapter object.

Declaration
// C#
public override DbDataAdapter CreateDataAdapter();

Return Value
A DbDataAdapter object that represents an OracleDataAdapter object.

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

CreateDataSourceEnumerator
This method returns a DbDataSourceEnumerator object that represents an OracleDataSourceEnumerator object.

Declaration
// C#
public override DbDataSourceEnumerator CreateDataSourceEnumerator();

Return Value
A DbDataSourceEnumerator object that represents an OracleDataSourceEnumerator object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Class
- OracleClientFactory Members
- "OracleDataSourceEnumerator Class" on page 7-36

CreateParameter
This method returns a DbParameter object that represents an OracleParameter object.

Declaration
// C#
public override DbParameter CreateParameter();
Return Value
A DbParameter object that represents an OracleParameter object.

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

CreatePermission
This method returns a CodeAccessPermission object that represents an OraclePermission object.

Declaration
// C#
    System.Security.Permissions.PermissionState state);

Parameter
- state
  A PermissionState object.

Return Value
A CodeAccessPermission object that represents an OraclePermission object.

Remarks
This method enables users, writing provider-independent code, to get a CodeAccessPermission instance that represents an OraclePermission object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleClientFactory Class
- OracleClientFactory Members
OracleConnectionStringBuilder Class

An OracleConnectionStringBuilder object allows applications to create or modify connection strings.

Supported Only in ADO.NET 2.0-Compliant ODP.NET

Class Inheritance
System.Object

Declaration
   // C#
   public sealed class OracleConnectionStringBuilder : DbConnectionStringBuilder

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

Remarks
The following rules must be followed for setting values with reserved characters:

1. Values containing characters enclosed within single quotes
   If the value contains characters that are enclosed within single quotation marks, then the entire value must be enclosed within double quotation marks.
   For example, password = "'scoTT'" where the value is 'scoTT'.

2. Values containing characters enclosed within double quotes
   Values should be enclosed in double quotation marks to preserve the case and to avoid the upper casing of values.
   If the value contains characters enclosed in double quotation marks, then it must be enclosed in single quotation marks.
   For example, password = "sco'TT" where the value is "scoTT".

3. Values containing characters enclosed in both single and double quotes
   If the value contains characters enclosed in both single and double quotation marks, the quotation mark used to enclose the value must be doubled each time it occurs within the value.
   For example, password = '"sco''TT"' where the value is "sco'TT".

4. Values containing spaces
   All leading and trailing spaces are ignored, but the spaces between the value are recognized. If the value needs to have leading or trailing spaces then it must be enclosed in double quotation marks.
   For example, User ID = Sco TT where the value is <Sco TT>.
   For example, User ID = "Sco TT " where the value is <Sco TT>. 
Keywords occurring multiple times in a connection string

If a specific keyword occurs multiple times in a connection string, the last occurrence listed is used in the value set.

For example, with "User ID = scott; password = tiger; User ID = david" connection string, User ID value is david.

Example

```csharp

class ConnectionStringBuilderSample
{
    static void Main(string[] args)
    {
        bool bRet = false;
        // Create an instance of OracleConnectionStringBuilder
        OracleConnectionStringBuilder connStrBuilder =
            new OracleConnectionStringBuilder();
        // Add new key/value pairs to the connection string
        connStrBuilder.Add("User Id", "scott");
        connStrBuilder.Add("Password", "tiger");
        connStrBuilder.Add("Data Source", "oracle");
        connStrBuilder.Add("pooling", false);
        // Modify the existing value
        connStrBuilder["Data source"] = "inst1";
        // Remove an entry from the connection string
        bRet = connStrBuilder.Remove("pooling");
        //ContainsKey indicates whether or not the specific key exist
        //returns true even if the user has not specified it explicitly
        Console.WriteLine("Enlist exist: " +
            connStrBuilder.ContainsKey("Enlist"));
        //returns false
        connStrBuilder.ContainsKey("Invalid");
        // ShouldSerialize indicates whether or not a specific key
        // exists in connection string inherited from DbConnectionStringBuilder.
        // returns true if the key is explicitly added the user otherwise false;
        // this will return false as this key doesn't exists.
        connStrBuilder.ShouldSerialize("user");
        // returns false because this key is not added by user explicitly.
        connStrBuilder.ShouldSerialize("Enlist");
        // IsFixedSize [read-only property]
        Console.WriteLine("Connection String is fixed size only: "
            + connStrBuilder.IsFixedSize);
        Console.WriteLine("Key/Value Pair Count: " + connStrBuilder.Count);
    }
}
```
//adding a new key which is not supported by the provider
//is not allowed.
try
{
    //this will throw an exception.
    connStrBuilder.Add("NewKey", "newValue");
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}

Console.WriteLine("Key/Value Pair Count: " + connStrBuilder.Count);

//modifying a existing key is allowed.
connStrBuilder.Add("Enlist", false);
Console.WriteLine("Key/Value Pair Count: " + connStrBuilder.Count);

// Get all the keys and values supported by the provider.
ICollection keyCollection = connStrBuilder.Keys;
ICollection valueCollection = connStrBuilder.Values;

IEnumerator keys = keyCollection.GetEnumerator();
IEnumerator values = valueCollection.GetEnumerator();

while (keys.MoveNext())
{
    values.MoveNext();
    Console.WriteLine("Key: {0}     Value: {1} 
" , keys.Current , values.Current);
}

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Members
- OracleConnectionStringBuilder Constructors
- OracleConnectionStringBuilder Public Properties
- OracleConnectionStringBuilder Public Methods
OracleConnectionStringBuilder Members

OracleConnectionStringBuilder members are listed in the following tables.

OracleConnectionStringBuilder Constructors
OracleConnectionStringBuilder constructors are listed in Table 7–5.

Table 7–5  OracleConnectionStringBuilder Constructors

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

OracleConnectionStringBuilder Public Properties
OracleConnectionStringBuilder instance properties are listed in Table 7–6.

Table 7–6  OracleConnectionStringBuilder Public Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrowsableConnectionString</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>ConnectionLifetime</td>
<td>Specifies the value corresponding to the Connection Lifetime attribute in the ConnectionString property</td>
</tr>
<tr>
<td>ConnectionString</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Specifies the value corresponding to the Connection Timeout attribute in the ConnectionString property</td>
</tr>
<tr>
<td>ContextConnection</td>
<td>Specifies the value corresponding to the Context Connection attribute in the ConnectionString property</td>
</tr>
<tr>
<td>Count</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>DataSource</td>
<td>Specifies the value corresponding to the Data Source attribute in the ConnectionString property</td>
</tr>
<tr>
<td>DBAPrivilege</td>
<td>Specifies the value corresponding to the DBA Privilege attribute in the ConnectionString property</td>
</tr>
<tr>
<td>DecrPoolSize</td>
<td>Specifies the value corresponding to the Decr Pool Size attribute in the ConnectionString property</td>
</tr>
<tr>
<td>Enlist</td>
<td>Specifies the value corresponding to the Enlist attribute in the ConnectionString property</td>
</tr>
<tr>
<td>HAEvents</td>
<td>Specifies the value corresponding to the HA Events attribute in the ConnectionString property</td>
</tr>
<tr>
<td>IncrPoolSize</td>
<td>Specifies the value corresponding to the Incr Pool Size attribute in the ConnectionString property</td>
</tr>
<tr>
<td>IsFixedSize</td>
<td>Indicates whether or not the Connection String Builder has a fixed size</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
</tbody>
</table>
### OracleConnectionStringBuilder Public Methods

OracleConnectionStringBuilder instance methods are listed in Table 7–7.

#### Table 7–7  OracleConnectionStringBuilder Public Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the connection string contents</td>
</tr>
</tbody>
</table>
### Table 7-7 (Cont.) OracleConnectionStringBuilder Public Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContainsKey</td>
<td>Indicates whether or not a specific attribute in the connection string is supported by ODP.NET</td>
</tr>
<tr>
<td>EquivalentTo</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes the entry corresponding to the specified attribute from the connection string</td>
</tr>
<tr>
<td>ShouldSerialize</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>TryGetValue</td>
<td>Returns the value corresponding to the supplied attribute, as an output parameter</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
OracleConnectionStringBuilder Constructors

OracleConnectionStringBuilder constructors instantiate new instances of the OracleConnectionStringBuilder class.

Overload List:

- **OracleConnectionStringBuilder()**
  This constructor instantiates a new instance of the OracleConnectionStringBuilder class.

- **OracleConnectionStringBuilder(string)**
  This constructor instantiates a new instance of the OracleConnectionStringBuilder class with the provided connection string.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members

OracleConnectionStringBuilder()

This constructor instantiates a new instance of the OracleConnectionStringBuilder class.

Declaration

```csharp
public OracleConnectionStringBuilder();
```

Remarks

The ConnectionString property is empty after the object is created.

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

OracleConnectionStringBuilder(string)

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

Declaration

```csharp
public OracleConnectionStringBuilder(string connectionString);
```

Parameters

- **connectionString**
  The connection information.

Exceptions

- ArgumentNullException - The connectionString parameter is null.
- ArgumentException - The connectionString parameter is invalid.
Remarks
The `ConnectionString` property of this instance is set to the supplied connection string.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnectionStringBuilder Class`
- `OracleConnectionStringBuilder Members`
**OracleConnectionStringBuilder Public Properties**

OracleConnectionStringBuilder public properties are listed in Table 7–8.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrowsableConnectionString</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>ConnectionLifetime</td>
<td>Specifies the value corresponding to the Connection Lifetime attribute in the ConnectionString property</td>
</tr>
<tr>
<td>ConnectionString</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Specifies the value corresponding to the Connection Timeout attribute in the ConnectionString property</td>
</tr>
<tr>
<td>ContextConnection</td>
<td>Specifies the value corresponding to the Context Connection attribute in the ConnectionString property</td>
</tr>
<tr>
<td>Count</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>DataSource</td>
<td>Specifies the value corresponding to the Data Source attribute in the ConnectionString property</td>
</tr>
<tr>
<td>DBAPrivilege</td>
<td>Specifies the value corresponding to the DBA Privilege attribute in the ConnectionString property</td>
</tr>
<tr>
<td>DecrPoolSize</td>
<td>Specifies the value corresponding to the Decr Pool Size attribute in the ConnectionString property</td>
</tr>
<tr>
<td>Enlist</td>
<td>Specifies the value corresponding to the Enlist attribute in the ConnectionString property</td>
</tr>
<tr>
<td>HAEvents</td>
<td>Specifies the value corresponding to the HA Events attribute in the ConnectionString property</td>
</tr>
<tr>
<td>IncrPoolSize</td>
<td>Specifies the value corresponding to the Incr Pool Size attribute in the ConnectionString property</td>
</tr>
<tr>
<td>IsFixedSize</td>
<td>Indicates whether or not the Connection String Builder has a fixed size</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>Item</td>
<td>Specifies the value associated with the specified attribute</td>
</tr>
<tr>
<td>Keys</td>
<td>Specifies a collection of attributes contained in the Connection String Builder</td>
</tr>
<tr>
<td>LoadBalancing</td>
<td>Specifies the value corresponding to the Load Balancing attribute in the ConnectionString property</td>
</tr>
<tr>
<td>MaxPoolSize</td>
<td>Specifies the value corresponding to the Max Pool Size attribute in the ConnectionString property</td>
</tr>
<tr>
<td>MetadataPooling</td>
<td>Specifies the value that corresponds to the Metadata Pooling attribute in the ConnectionString property</td>
</tr>
<tr>
<td>MinPoolSize</td>
<td>Specifies the value corresponding to the Min Pool Size attribute in the ConnectionString property</td>
</tr>
</tbody>
</table>
**OracleConnectionStringBuilder Class**

**ADO.NET 2.0 Classes**

---

### Table 7–8 (Cont.) OracleConnectionStringBuilder Public Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Specifies the value corresponding to the Password attribute in the ConnectionString property</td>
</tr>
<tr>
<td>PersistSecurityInfo</td>
<td>Specifies the value corresponding to the Persist Security Info attribute in the ConnectionString property</td>
</tr>
<tr>
<td>Pooling</td>
<td>Specifies the value corresponding to the Pooling attribute in the ConnectionString property</td>
</tr>
<tr>
<td>ProxyPassword</td>
<td>Specifies the value corresponding to the Proxy User Id attribute in the ConnectionString property</td>
</tr>
<tr>
<td>ProxyUserId</td>
<td>Specifies the value corresponding to the Proxy User Id attribute in the ConnectionString property</td>
</tr>
<tr>
<td>SelfTuning</td>
<td>Specifies the value corresponding to the Self Tuning attribute in the ConnectionString property</td>
</tr>
<tr>
<td>StatementCachePurge</td>
<td>Specifies the value corresponding to the Statement Cache Purge attribute in the ConnectionString property</td>
</tr>
<tr>
<td>StatementCacheSize</td>
<td>Specifies the value corresponding to the Statement Cache Size attribute in the ConnectionString property</td>
</tr>
<tr>
<td>UserID</td>
<td>Specifies the value corresponding to the User Id attribute in the ConnectionString property</td>
</tr>
<tr>
<td>ValidateConnection</td>
<td>Specifies the value corresponding to the Validate Connection attribute in the ConnectionString property</td>
</tr>
<tr>
<td>Values</td>
<td>Specifies a collection of values contained in the Connection String Builder</td>
</tr>
</tbody>
</table>

---

### See Also:

- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members

---

### ConnectionLifetime

This property specifies the value corresponding to the Connection Lifetime attribute in the ConnectionString property.

**Declaration**

```csharp
// C#
public int ConnectionLifetime{get; set;}
```

---

### Property Value

An int that represents the value of the supplied attribute.

---

### Exceptions

OracleException - The specified value is less than zero.
Remarks
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

SqlConnection
This property specifies the value corresponding to the Connection Timeout attribute in the ConnectionString property.

Declaration
// C#
public int ConnectionTimeout {get; set;}

Property Value
An int that represents the value of the supplied attribute.

Exceptions
OracleException - The specified value is less than zero.

Remarks
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

ContextConnection
This property specifies the value corresponding to the Context Connection attribute in the ConnectionString property.

Declaration
// C#
public bool ContextConnection {get; set;}

Property Value
A bool that represents the value of the supplied attribute.
OracleConnectionStringBuilder Class

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

**DataSource**

This property specifies the value corresponding to the DataSource attribute in the ConnectionString property.

**Declaration**

```csharp
// C#
public string DataSource { get; set; }
```

**Property Value**

A string that represents the value of the supplied attribute.

**Exceptions**

- ArgumentNullException - The specified value is null.

**Remarks**

When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

**DBAPrivilege**

This property specifies the value corresponding to the DBA Privilege attribute in the ConnectionString property.

**Declaration**

```csharp
// C#
public string DBAPrivilege { get; set; }
```

**Property Value**

A string that represents the value of the supplied attribute.

Possible values are SYSDBA or SYSOPER.

**Exceptions**

- ArgumentNullException - The specified value is null.
- OracleException - The specified value is invalid.
Remarks
When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnectionStringBuilder` Class
- `OracleConnectionStringBuilder` Members

**DecrPoolSize**

This property specifies the value corresponding to the `Decr Pool Size` attribute in the `ConnectionString` property.

Declaration

```csharp
// C#
public int DecrPoolSize{get; set;}
```

Property Value
An `int` that represents the value of the supplied attribute.

Exceptions

- `OracleException` - The specified value is less than 1.

Remarks

When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnectionStringBuilder` Class
- `OracleConnectionStringBuilder` Members

**Enlist**

This property specifies the value corresponding to the `Enlist` attribute in the `ConnectionString` property.

Declaration

```csharp
// C#
public string Enlist{get; set;};
```

Property Value
A string that represents the value of the supplied attribute. Values are case-insensitive. Possible values are: dynamic, true, false, yes, and no.

Exceptions

- `ArgumentNullException` - The specified value is null.
OracleException - The supplied value is not one of following: dynamic, true, false, yes, or no.

Remarks
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

HAEvents
This property specifies the value corresponding to the HAEvents attribute in the ConnectionString property.

Declaration
// C#
public bool HAEvents{get; set;}

Property Value
A bool that represents the value of the supplied attribute.

Remarks
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

IncrPoolSize
This property specifies the value corresponding to the IncrPoolSize attribute in the ConnectionString property.

Declaration
// C#
public int IncrPoolSize{get; set;}

Property Value
An int that represents the value of the supplied attribute.

Exceptions
OracleException - The specified value is less than 1.
OracleConnectionStringBuilder Public Properties

**Remarks**
When an `OracleConnectionStringBuilder` instance is created, this property is set to the default value of the corresponding connection string attribute.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnectionStringBuilder` Class
- `OracleConnectionStringBuilder` Members

**IsFixedSize**
Indicates whether or not the Connection String Builder has a fixed size.

**Declaration**
```
// C#
public override bool IsFixedSize{get;}
```

**Property Value**
Returns `true` if the Connection String Builder has a fixed size; otherwise, returns `false`.

**Remarks**
Attributes cannot be added or removed. They can only be modified for connection strings with a fixed size.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleConnectionStringBuilder` Class
- `OracleConnectionStringBuilder` Members

**Item**
This property specifies the value associated with the specified attribute.

**Declaration**
```
// C#
public override object this[string keyword]{get; set;}
```

**Property Value**
An object value corresponding to the attribute.

**Exceptions**
- `ArgumentNullException` - The specified attribute is null.
- `OracleException` - The specified attribute is not supported or the specified value is invalid.
OracleConnectionStringBuilder Class

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

Keys

This property specifies a collection of attributes contained in the Connection String Builder.

Declaration
// C#
public override ICollection Keys{get;}

Property Value
Returns an ICollection that represents the attributes in the Connection String Builder.

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

LoadBalancing

This property specifies the value corresponding to the Load Balancing attribute in the ConnectionString property.

Declaration
// C#
public bool LoadBalancing {get; set;}

Property Value
A bool that contains the value of the supplied attribute.

Remarks
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

MaxPoolSize

This property specifies the value corresponding to the Max Pool Size attribute in the ConnectionString property.
OracleConnectionStringBuilder Public Properties

Declaration

// C#
public int MaxPoolSize {get; set;}

Property Value
An int that represents the value of the supplied attribute.

Exceptions
OracleException - The specified value is less than 1.

Remarks
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

MetadataPooling
This property specifies the value that corresponds to the Metadata Pooling attribute in the ConnectionString property.

Declaration

// C#
public bool MetadataPooling {get; set;};

Property Value
A bool containing the value of the supplied attribute.

Remarks
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

MinPoolSize
This property specifies the value corresponding to the Min Pool Size attribute in the ConnectionString property.

Declaration

// C#
public int MinPoolSize {get; set;}

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

**Property Value**
An int that contains the value of the supplied attribute.

**Exceptions**
OracleException - The specified value is less than 0.

**Remarks**
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

**Password**
This property specifies the value corresponding to the Password attribute in the ConnectionString property.

**Declaration**
// C#
public string Password { get; set; }

**Property Value**
A string that contains the value of the supplied attribute.

**Exception**
ArgumentNullException - The specified value is null.

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

**PersistSecurityInfo**
This property specifies the value corresponding to the Persist Security Info attribute in the ConnectionString property.

**Declaration**
// C#
public bool PersistSecurityInfo { get; set; }

**Property Value**
A bool that represents the value of the supplied attribute.
Remarks
When an OracleConnectionStringBuilder instance is created, this property gets set to the default value of the corresponding connection string attribute.

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

Pooling
This property specifies the value corresponding to the Pooling attribute in the ConnectionString property.

Declaration
// C#
public bool Pooling {get; set;}

Property Value
A bool that represents the value of the supplied attribute.

Remarks
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

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

ProxyPassword
This property specifies the value corresponding to the Proxy Password attribute in the ConnectionString property.

Declaration
// C#
public string ProxyPassword {get; set;}

Property Value
A string that represents the value of the supplied attribute.

Exception
ArgumentNullException - The specified value is null.
ProxyUserId
This property specifies the value corresponding to the Proxy UserId attribute in the ConnectionString property.

Declaration
// C#
public string ProxyUserId {get; set;}

Property Value
A string that represents the value of the supplied attribute.

Exception
ArgumentNullException - The specified value is null.

See Also:
■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleConnectionStringBuilder Class
■ OracleConnectionStringBuilder Members

SelfTuning
This property specifies the value corresponding to the Self Tuning attribute in the ConnectionString property.

Declaration
// C#
public bool SelfTuning {get; set;}

Property Value
A bool that represents the value of the supplied attribute.

See Also:
■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleConnectionStringBuilder Class
■ OracleConnectionStringBuilder Members

StatementCachePurge
This property specifies the value corresponding to the Statement Cache Purge attribute in the ConnectionString property.

Declaration
// C#
public bool StatementCachePurge {get; set;}

**Property Value**
A bool that represents the value of the supplied attribute.

**Remarks**
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members

**StatementCacheSize**
This property specifies the value corresponding to the Statement Cache Size attribute in the ConnectionString property.

**Declaration**
```csharp
// C#
public int StatementCacheSize{get; set;}
```

**Property Value**
An int that represents the value of the supplied attribute.

**Exceptions**
OracleException - The specified value is less than zero.

**Remarks**
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members

**UserID**
This property specifies the value corresponding to the User Id attribute in the ConnectionString property.

**Declaration**
```csharp
// C#
public string UserID{get; set;}
```
**Property Value**
A string that represents the value of the supplied attribute.

**Exception**
ArgumentNullException - The specified value is null.

**ValidateConnection**
This property specifies the value corresponding to the Validate Connection attribute in the ConnectionString property.

**Declaration**
// C#
public bool ValidateConnection{get; set;}

**Property Value**
A bool that represents the value of the supplied attribute.

**Remarks**
When an OracleConnectionStringBuilder instance is created, this property is set to the default value of the corresponding connection string attribute.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members

**Values**
This property specifies a collection of values contained in the Connection String Builder.

**Declaration**
// C#
public override ICollection Values{get;}

**Property Value**
Returns an ICollection that represents the values in the Connection String Builder.

**Remarks**
The order of the values in the ICollection is unspecified, but is the same as the associated attributes in the ICollection returned by the Keys property.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members
OracleConnectionStringBuilder Public Methods

OracleConnectionStringBuilder public methods are listed in Table 7–9.

Table 7–9  OracleConnectionStringBuilder Public Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the connection string contents</td>
</tr>
<tr>
<td>ContainsKey</td>
<td>Indicates whether or not a specific attribute in the connection string is supported by ODP.NET</td>
</tr>
<tr>
<td>EquivalentTo</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes the entry corresponding to the specified attribute from the connection string</td>
</tr>
<tr>
<td>ShouldSerialize</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Data.Common.DbConnectionStringBuilder</td>
</tr>
<tr>
<td>TryGetValue</td>
<td>Returns the value corresponding to the supplied attribute, as an output parameter</td>
</tr>
</tbody>
</table>

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

Clear

This method clears the connection string contents.

Declaration

```csharp
// C#
public override void Clear();
```

Remarks

All key/value pairs are removed from the OracleConnectionStringBuilder object and the ConnectionString property is set to Empty.

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

ContainsKey

This method indicates whether or not a specific attribute in the connection string is supported by ODP.NET.
Declaration

// C#
public override bool ContainsKey(string keyword);

Parameters

- **keyword**
  The attribute being verified.

Return Value

Returns `true` if the specified attribute exists; otherwise, returns `false`.

Exceptions

- **ArgumentNullException** - The specified attribute is null.

Remarks

This method indicates if the attribute is part of the provider-supported attributes. It does not indicate if the user added the attribute to the connection string.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members

Remove

This method removes the entry corresponding to the specified attribute from the connection string.

Declaration

// C#
public override bool Remove(string keyword);

Parameters

- **keyword**
  The attribute that specifies the entry to be removed.

Return Value

Returns `true` if the attribute existed in the connection string and the corresponding entry was removed; otherwise, returns `false`.

Exceptions

- **ArgumentNullException** - The specified attribute is null.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members
TryGetValue

This method returns the value corresponding to the supplied attribute, as an output parameter.

**Declaration**

```
// C#
public override bool TryGetValue(string keyword, out object value);
```

**Parameters**

- `keyword`  
The attribute for which the value is being retrieved.
- `value`  
The value of the supplied attribute.
  
  Sets value to the default value if the attribute is not present in the connection string.

**Return Value**

Returns `true` if the value that corresponds to the attribute has been successfully retrieved; otherwise, returns `false`. If the attribute is not present in the connection string, returns `false` and sets the `value` to null.

**Exceptions**

- `ArgumentNullException` - The specified attribute is null.

**Remarks**

If the function returns `false`, sets `value` to null.

If the attribute is not present in the connection string, sets `value` to the default value.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleConnectionStringBuilder Class
- OracleConnectionStringBuilder Members
OracleDataSourceEnumerator Class

An OracleDataSourceEnumerator object allows applications to generically obtain a collection of data sources to connect to.

Supported Only in ADO.NET 2.0-Compliant ODP.NET

Class Inheritance

System.Object

    System.DbDataSourceEnumerator


Declaration

    // C#
    public sealed class OracleDataSourceEnumerator : DbDataSourceEnumerator

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

    class DataSourceEnumSample
    {
        static void Main()
        {
            string ProviderName = "Oracle.DataAccess.Client";
            DbProviderFactory factory = DbProviderFactories.GetFactory(ProviderName);

            if (factory.CanCreateDataSourceEnumerator)
            {
                OracleDataSourceEnumerator dsenum = factory.CreateDataSourceEnumerator();
                DataTable dt = dsenum.GetDataSources();

                // Print the first column/row entry in the DataTable
                Console.WriteLine(dt.Columns[0] + " : " + dt.Rows[0][0]);
                Console.WriteLine(dt.Columns[1] + " : " + dt.Rows[0][1]);
            }
            else
            {
                Console.Write("Data source enumeration is not supported by provider");
            }
        }
    }
Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataSourceEnumerator Members
- OracleDataSourceEnumerator Public Methods
OracleDataSourceEnumerator Members

OracleDataSourceEnumerator members are listed in the following tables.

OracleDataSourceEnumerator Public Methods
OracleDataSourceEnumerator Public Methods are listed in Table 7–10.

Table 7–10 OracleDataSourceEnumerator Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDataSources</td>
<td>Returns a DataTable object with information on all the TNS alias entries in the tnsnames.ora file</td>
</tr>
</tbody>
</table>

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

OracleDataSourceEnumerator Public Methods

The OracleDataSourceEnumerator static method is listed in Table 7-11.

Table 7-11  OracleDataSourceEnumerator Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDataSources</td>
<td>Returns a DataTable object with information on all the TNS alias entries in the tnsnames.ora file</td>
</tr>
</tbody>
</table>

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

GetDataSources

This method returns a DataTable object with information on all the TNS alias entries in the tnsnames.ora file.

Declaration

```csharp
// C#
public override DataTable GetDataSources();
```

Return Value

A DataTable object.

Remarks

This method returns a DataTable object for each TNS Alias entry that exists in the tnsnames.ora file.

If the tnsnames.ora file is not found, then the returned DataTable object is empty.

The following columns are returned for each row, but only the InstanceName column is populated.

- InstanceName (type: System.String)
- ServerName (type: System.String)
- ServiceName (type: System.String)
- Protocol (type: System.String)
- Port (type: System.String)

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDataSourceEnumerator Class
- OracleDataSourceEnumerator Members
This chapter describes the following ODP.NET HA event class and enumerations:

- OracleHAEventArgs Class
- OracleHAEventHandler Delegate
- OracleHAEventSource Enumeration
- OracleHAEventStatus Enumeration
OracleHAEventArgs Class

The OracleHAEventArgs class provides event data for the OracleConnection.HAEvent event.

Class Inheritance

System.Object

  System.EventArgs


Declaration

// C#
public sealed class OracleHAEventArgs

Thread Safety

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

Remarks

When any HA event occurs for a service, service member, host, node, or instance that an OracleConnection object is set to with "ha events=true", the OracleConnection.HAEvent is triggered and passes an instance of OracleHAEventArgs to all the delegates that have registered with the event.

Requirements

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Members
- OracleHAEventArgs Properties
- "OracleConnection Class" on page 5-64
- "HAEvent" on page 5-113
OracleHAEventArgs Members

OracleHAEventArgs members are listed in the following table.

**OracleHAEventArgs Properties**

The OracleHAEventArgs properties are listed in Table 8–2.

**Table 8–1 OracleHAEventArgs Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatabaseDomainName</td>
<td>Specifies the domain name of the database affected by the HA event</td>
</tr>
<tr>
<td>DatabaseName</td>
<td>Specifies the database affected by the HA event</td>
</tr>
<tr>
<td>HostName</td>
<td>Specifies the host that triggered the event</td>
</tr>
<tr>
<td>InstanceName</td>
<td>Specifies the instance that triggered the event</td>
</tr>
<tr>
<td>ServiceName</td>
<td>Specifies the service that triggered the event</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the source that triggered the event</td>
</tr>
<tr>
<td>Status</td>
<td>Specifies the status of the source that triggered the event</td>
</tr>
<tr>
<td>Time</td>
<td>Specifies the time when the event was triggered on the server</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- "HAEvent" on page 5-113
OracleHAEventArgs Properties

The OracleHAEventArgs properties are listed in Table 8–2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatabaseDomainName</td>
<td>Specifies the domain name of the database affected by the HA event.</td>
</tr>
<tr>
<td>DatabaseName</td>
<td>Specifies the database affected by the HA event.</td>
</tr>
<tr>
<td>HostName</td>
<td>Specifies the host that triggered the event.</td>
</tr>
<tr>
<td>InstanceName</td>
<td>Specifies the instance that triggered the event.</td>
</tr>
<tr>
<td>ServiceName</td>
<td>Specifies the service that triggered the event.</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the source that triggered the event.</td>
</tr>
<tr>
<td>Status</td>
<td>Specifies the status of the source that triggered the event.</td>
</tr>
<tr>
<td>Time</td>
<td>Specifies the time when the event was triggered on the server.</td>
</tr>
</tbody>
</table>

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

DatabaseDomainName

This property specifies the domain name of the database that is affected by the HA event.

Declaration

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

Property Value

The domain name of the database that is affected by the HA Event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- OracleHAEventArgs Members
- "HAEvent" on page 5-113

DatabaseName

This property specifies the database that is affected by the HA event.

Declaration

```csharp
// C#
public string DatabaseName {get;}
```
Property Value
This property specifies the database name that is affected by the HA event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- OracleHAEventArgs Members
- "HAEvent" on page 5-113

HostName
This property specifies the host that triggered the HA event.

Declaration
// C#
public string HostName {get;}

Property Value
The host that is affected by the HA Event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- OracleHAEventArgs Members
- "HAEvent" on page 5-113

InstanceName
This property specifies the instance that triggered the HA event.

Declaration
// C#
public string InstanceName {get;}

Property Value
The instance that is affected by the HA Event.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- OracleHAEventArgs Members
- "HAEvent" on page 5-113

ServiceName
This property specifies the service that triggered the HA event.
OracleHAEventArgs Properties

Declaration

// C#
public string ServiceName {get;}

Property Value
The service that is affected by the HA Event.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- OracleHAEventArgs Members
- "HAEvent" on page 5-113

Source
This property specifies the source that triggered the HA event.

Declaration

// C#
public OracleHAEventSource Source {get;}

Property Value
The source that triggered the HA Event.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- OracleHAEventArgs Members
- "HAEvent" on page 5-113

Status
This property specifies the status of the source that triggered the HA event.

Declaration

// C#
public OracleHAEventStatus Status {get;}

Property Value
The status of the source that triggered the HA Event.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- OracleHAEventArgs Members
- "HAEvent" on page 5-113
Time

This property specifies the time when the HA event was triggered on the server.

**Declaration**

```csharp
// C#
public DateTime Time {get;}
```

**Property Value**

The time that the HA Event was triggered.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- OracleHAEventArgs Members
- "HAEvent" on page 5-113
OracleHAEventHandler Delegate

The OracleHAEventHandler delegate represents the signature of the method that handles the OracleConnection.HAEvent event.

Declaration

// C#
public delegate void OracleHAEventHandler(object sender, OracleHAEventArgs eventArgs);

Parameters

- sender
  The source of the event.
- EventArgs
  The OracleHAEventArgs object that contains the event data.

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll


See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- "HAEvent" on page 5-113
OracleHAEventSource Enumeration

The OracleHAEventSource enumeration indicates the source of the HA event. Table 8–3 lists all the OracleHAEventSource enumeration values with a description of each enumerated value.

Table 8–3  OracleHAEventSource Enumeration Member Values

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>The source of the HA Event is a service.</td>
</tr>
<tr>
<td>ServiceMember</td>
<td>The source of the HA Event is a service member.</td>
</tr>
<tr>
<td>Database</td>
<td>The source of the HA Event is a database.</td>
</tr>
<tr>
<td>Host</td>
<td>The source of the HA Event is a host.</td>
</tr>
<tr>
<td>Instance</td>
<td>The source of the HA Event is an instance.</td>
</tr>
</tbody>
</table>

Requirements

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleHAEventArgs Class
- "Source" on page 8-6
OracleHAEventStatus Enumeration

The `OracleHAEventStatus` enumeration indicates the status of the HA event source.

Table 8–4 lists all the `OracleHAEventStatus` enumeration values with a description of each enumerated value.

### Table 8–4  `OracleHAEventStatus` Enumeration Values

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>The source of the HA Event is up.</td>
</tr>
<tr>
<td>Down</td>
<td>The source of the HA Event is down.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`


See Also:

- "`Oracle.DataAccess.Client Namespace`" on page 1-4
- `OracleHAEventArgs Class`
- "`Status`" on page 8-6
This chapter describes Oracle Data Provider for .NET Change Notification Classes, Event Delegates, and Enumerations, which support Continuous Query Notification.

**Note:** Database Change Notification is known as Continuous Query Notification in Oracle database documentation.

**See Also:** "Database Change Notification Support" on page 3-120

This chapter contains these topics:

- OracleDependency Class
- OracleNotificationRequest Class
- OracleNotificationEventArgs Class
- OnChangeEventHandler Delegate
- OracleRowidInfo Enumeration
- OracleNotificationType Enumeration
- OracleNotificationSource Enumeration
- OracleNotificationInfo Enumeration
OracleDependency Class

An OracleDependency class represents a dependency between an application and an Oracle database, enabling the application to get notifications whenever the data of interest or the state of the Oracle database changes.

Class Inheritance
System.Object


Declaration
// C#
public sealed class OracleDependency

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

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
Comment: Not supported in a .NET stored procedure

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Members
- OracleDependency Constructors
- OracleDependency Static Fields
- OracleDependency Static Methods
- OracleDependency Methods
- OracleDependency Properties
- OracleDependency Events
OracleDependency Members

OracleDependency members are listed in the following tables.

OracleDependency Constructors
OracleDependency constructors are listed in Table 9–1.

Table 9–1 OracleDependency Constructors

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

OracleDependency Static Fields
The OracleDependency static field is listed in Table 9–2.

Table 9–2 OracleDependency Static Field

<table>
<thead>
<tr>
<th>Static Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Indicates the port number that the notification listener listens on, for database notifications</td>
</tr>
</tbody>
</table>

OracleDependency Static Methods
OracleDependency static methods are listed in Table 9–3.

Table 9–3 OracleDependency Static Methods

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetOracleDependency</td>
<td>Returns an OracleDependency instance based on the specified unique identifier</td>
</tr>
</tbody>
</table>

OracleDependency Properties
OracleDependency properties are listed in Table 9–4.

Table 9–4 OracleDependency Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource</td>
<td>Indicates the data source associated with the OracleDependency instance</td>
</tr>
<tr>
<td>HasChanges</td>
<td>Indicates whether or not there is any change in the database associated with this dependency</td>
</tr>
<tr>
<td>Id</td>
<td>Represents the unique identifier for the OracleDependency instance</td>
</tr>
<tr>
<td>IsEnabled</td>
<td>Specifies whether or not the dependency is enabled between the application and the database</td>
</tr>
<tr>
<td>QueryBasedNotification</td>
<td>Specifies whether the change notification registration is object-based or query-based</td>
</tr>
<tr>
<td>RegisteredQueryIDs</td>
<td>Provides a list of CHANGE_NOTIFICATION_QUERY_IDS</td>
</tr>
<tr>
<td>RegisteredResources</td>
<td>Indicates the database resources that are registered in the notification registration</td>
</tr>
</tbody>
</table>
OracleDependency Members

Table 9–4 (Cont.) OracleDependency Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowIdInfo</td>
<td>Specifies whether or not ROWID information is part of change notification events fired whenever data changes on the database</td>
</tr>
<tr>
<td>UserName</td>
<td>Indicates the database user name associated with the OracleDependency instance</td>
</tr>
</tbody>
</table>

OracleDependency Methods

OracleDependency methods are listed in Table 9–5.

Table 9–5 OracleDependency Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCommandDependency</td>
<td>Binds the OracleDependency instance to the specified OracleCommand instance</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>RemoveRegistration</td>
<td>Removes the specified dependency between the application and the database</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

OracleDependency Events

The OracleDependency event is listed in Table 9–6.

Table 9–6 OracleDependency Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnChange</td>
<td>An event that is sent when a database notification associated with the dependency is received from the database</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
OracleDependency Constructors

OracleDependency constructors create instances of the OracleDependency class.

Overload List:

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

- **OracleDependency(OracleCommand)**
  This constructor creates an instance of the OracleDependency class and binds it to the specified OracleCommand instance.

- **OracleDependency(OracleCommand, bool, int, bool)**
  This constructor creates an instance of the OracleDependency class and binds it to the specified OracleCommand instance, specifying whether or not a notification is to be removed upon notification, the timeout value of the notification registration, and the persistence of the notification.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members

**OracleDependency ()**

This constructor creates an instance of the OracleDependency class.

Declaration

```csharp
// C#
public OracleDependency ()
```

Remarks

Using this constructor does not bind any OracleCommand to the newly constructed OracleDependency. Use the AddCommandDependency method to do so.

---

**Note:** The dependency between the application and the database is not established when the OracleDependency instance is created. The dependency is established when the command that is associated with this dependency is executed.

---

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members

**OracleDependency(OracleCommand)**

This constructor creates an instance of the OracleDependency class and binds it to an OracleCommand instance.
OracleDependency Constructors

Declaration

// C#
public OracleDependency {OracleCommand cmd}

Parameters

■ cmd

The command that the OracleDependency object binds to.

Exceptions

ArgumentNullException - The cmd parameter is null.

InvalidOperationException - The specified OracleCommand instance already contains a notification request.

Remarks

When this OracleDependency constructor binds the OracleCommand instance to an OracleDependency instance, it causes the creation of an OracleNotificationRequest instance and then sets that OracleNotificationRequest instance to the OracleCommand.Notification property.

The Continuous Query Notification is registered with the database, when the command is executed. Any of the command execution methods (for example, ExecuteNonQuery, ExecuteReader, and so on) will register the notification request. An OracleDependency may be bound to more than one OracleCommand. When one of these OracleCommand object statements is executed, the statement is registered with the associated OracleCommand. Although the registration happens on each OracleCommand separately, one OracleDependency can be responsible for detecting and sending notifications that occur for all OracleCommand objects that the OracleDependency is associated with. The OnChangeEventArgs that is passed to the application for the OnChange event provides information on what has changed in the database.

The OracleNotificationRequest instance that is created by this constructor has the following default property values:

■ IsNotifiedOnce is set to the value True.

■ Timeout is set to 50,000 seconds.

■ IsPersistent is set to the value False, that is, the invalidation message is not persistent, but is stored in an in-memory queue before delivery.

See Also:

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

■ OracleDependency Class

■ OracleDependency Members

OracleDependency(OracleCommand, bool, int, bool)

This constructor creates an instance of the OracleDependency class and binds it to the specified OracleCommand instance, while specifying whether or not a registration is to be removed upon notification, the timeout value of the notification registration, and the persistence of the notification.
Declaration

```csharp
// C#
public OracleDependency (OracleCommand cmd, bool isNotifiedOnce, long timeout, bool isPersistent)
```

Parameters

- **cmd**
  The command associated with the Continuous Query Notification request.

- **isNotifiedOnce**
  An indicator that specifies whether or not the registration is removed automatically once the notification occurs.

- **timeout**
  The amount of time, in seconds, that the registration stays active. When `timeout` is set to 0, the registration never expires. The valid values for `timeout` are between 0 and 4294967295.

- **isPersistent**
  Indicates whether or not the invalidation message should be queued persistently in the database before delivery. If the `isPersistent` parameter is set to `true`, the message is queued persistently in the database and cannot be lost upon database failures or shutdowns. If the `isPersistent` property is set to `false`, the message is stored in an in-memory queue before delivery and might be lost. Database performance is faster if the message is stored in an in-memory queue rather than in the database queue.

Exceptions

- **ArgumentNullException** - The `cmd` parameter is null.

- **ArgumentOutOfRangeException** - The specified `timeout` is invalid.

- **InvalidOperationException** - The specified `OracleCommand` instance already contains a notification request.

Remarks

When this OracleDependency constructor binds the `OracleCommand` instance to an OracleDependency instance, it causes the creation of an OracleNotificationRequest instance and then sets that OracleNotificationRequest instance to the `OracleCommand.Notification` property.

The Continuous Query Notification is registered with the database, when the command is executed. Any of the command execution methods (for example, `ExecuteNonQuery`, `ExecuteReader`, and so on) will register the notification request. An OracleDependency may be bound to more than one `OracleCommand`. When one of these `OracleCommand` object statements is executed, the statement is registered with the associated `OracleCommand`. Although the registration happens on each `OracleCommand` separately, one `OracleDependency` can be responsible for detecting and sending notifications that occur for all `OracleCommand` objects that the OracleDependency is associated with. The `OnChangeEventArgs` that is passed to the application for the `OnChange` event provides information on what has changed in the database.
The `OracleNotificationRequest` instance that is created by this constructor has the following default property values:

- `IsNotifiedOnce` is set to the specified value.
- `Timeout` is set to the specified value.
- `IsPersistent` is set to the specified value.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleDependency Class`
- `OracleDependency Members`
OracleDependency Static Fields

The `OracleDependency` static field is listed in Table 9–7.

<table>
<thead>
<tr>
<th>Static Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Indicates the port number that the notification listener listens on, for database notifications</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleDependency Class`
- `OracleDependency Members`

Port

This static field indicates the port number that the notification listener listens on, for database notifications.

Declaration

```
// C#
public static int Port { get; set; }
```

Property Value

An `int` value that represents the number of the port that listens for the database notifications. If the port number is set to -1, a random port number is assigned for the listener when the listener is started. Otherwise, the specified port number is used to start the listener.

Exceptions

- `ArgumentOutOfRangeException` - The port number is set to a negative value.
- `InvalidOperationException` - The port number is being changed after the listener has started.

Remarks

The port number specified by the `OracleDependency.Port` static field is used by the notification listener that runs within the same application domain as ODP.NET. This port number receives Continuous Query Notifications from the database. One notification listener is capable of listening to all Continuous Query Notifications and therefore, only one notification listener is created for each application domain.

The notification listener is created when a command associated with an `OracleDependency` object is executed for the first time during the application domain lifetime. The port number specified for the `OracleDependency.Port` static field is used by the listener for its lifetime. The `OracleDependency.Port` static field can be changed after the creation of the notification listener, but doing so does not affect the actual port number that the notification listener listens on.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members
OracleDependency Static Methods

OracleDependency static methods are listed in Table 9–8.

Table 9–8 OracleDependency Static Methods

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetOracleDependency</td>
<td>Returns an OracleDependency instance based on the specified unique identifier</td>
</tr>
</tbody>
</table>

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

GetOracleDependency

This static method returns an OracleDependency instance based on the specified unique identifier.

Declaration

```csharp
// C#
public static OracleDependency GetOracleDependency(string guid)
```

Parameters

- `guid`
  The string representation of the unique identifier of an OracleDependency instance.

Exceptions

- ArgumentException - The specified unique identifier cannot locate an OracleDependency instance.

Return Value

An OracleDependency instance that has the specified `guid` parameter.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members
OracleDependency Properties

OracleDependency properties are listed in Table 9–9.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource</td>
<td>Indicates the data source associated with the OracleDependency instance</td>
</tr>
<tr>
<td>HasChanges</td>
<td>Indicates whether or not there is any change in the database associated with this dependency</td>
</tr>
<tr>
<td>Id</td>
<td>Represents the unique identifier for the OracleDependency instance</td>
</tr>
<tr>
<td>IsEnabled</td>
<td>Specifies whether or not the dependency is enabled between the application and the database</td>
</tr>
<tr>
<td>QueryBasedNotification</td>
<td>Specifies whether the change notification registration is object-based or query-based</td>
</tr>
<tr>
<td>RegisteredQueryIDs</td>
<td>Provides a list of CHANGE_NOTIFICATION_QUERY_IDS</td>
</tr>
<tr>
<td>RegisteredResources</td>
<td>Indicates the database resources that are registered in the notification registration</td>
</tr>
<tr>
<td>RowidInfo</td>
<td>Specifies whether or not ROWID information is part of change notification events fired whenever data changes on the database</td>
</tr>
<tr>
<td>UserName</td>
<td>Indicates the database user name associated with the OracleDependency instance</td>
</tr>
</tbody>
</table>

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

DataSource

This property indicates the data source associated with the OracleDependency instance.

Declaration

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

Property Value

A string that indicates the data source associated with the OracleDependency instance.

Remarks

The DataSource property is populated with the data source once the OracleCommand associated with the OracleDependency executes and registers for the notification successfully.
HasChanges

This property indicates whether or not there is any change in the database associated with this dependency.

**Declaration**

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

**Property Value**

A `bool` value that returns `True` if the database has detected changes that are associated with this dependency; otherwise, returns `False`.

**Remarks**

As an alternative to using the `OnChange` event, applications can check the `HasChanges` property to determine if there are any changes in the database associated with this dependency.

Once the `HasChanges` property is accessed, its value is reset to `False` so that the next notification can then be acknowledged.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleDependency Class`
- `OracleDependency Members`

Id

This property represents the unique identifier for the `OracleDependency` instance.

**Declaration**

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

**Property Value**

A string that represents the unique identifier that was generated for the `OracleDependency` instance when it was created.

**Remarks**

This property is set when the `OracleDependency` instance is created.
OracleDependency Properties

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

IsEnabled
This property specifies whether or not the dependency is enabled between the application and the database.

Declaration

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

Property Value
A `bool` value that specifies whether or not dependency is enabled between the application and the database.

Remarks
The dependency between the application and the database is not established when the `OracleDependency` instance is created. The dependency is established when the command that is associated with this dependency is executed, at which time the notification request is registered with the database. The dependency ends when the notification registration is removed from the database or when it times out.

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

QueryBasedNotification
This instance property specifies whether the change notification registration is object-based or query-based.

Declaration

```csharp
// C#
public bool QueryBasedNotification{get; set;}
```

Property Value
Specifies whether the change notification registration is object-based or not.

Exceptions
`ObjectDisposedException` - The object is already disposed.

Remarks
This property value will be ignored if it is set after the command execution that registers the command for change notification.

By default, this property is true.
ODP.NET developers can register their queries on the row level, not just the object level, beginning with Oracle Data Provider for .NET release 11.1 and Oracle Database 11g release 1 (11.1). The application only receives notification when the selected row or rows change. Query-based notifications provide developers more granularity for using client-side cached data, as they can be more specific about what changes the application needs to be notified of.

OracleNotificationType enumeration is set to Query for query-based notifications.

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

RegisteredQueryIDs
This instance property provides a list of CHANGE_NOTIFICATION_QUERY_IDS.

Declaration

```csharp
// C#
public ArrayList RegisteredQueryIDs {get;}
```

Property Value
This property is an ArrayList of CHANGE_NOTIFICATION_QUERY_IDS.

Exceptions
ObjectDisposedException - The object is already disposed.

Remarks
This property provides a list of CHANGE_NOTIFICATION_QUERY_IDS that uniquely identify the query that has been registered for change notification. The notification returned from the database will also contain these IDs, allowing applications to determine the query that the notifications are for.

The QueryId at index n in RegisteredQueryIDs is for the statement at index n the RegisteredResources at index n.

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

RegisteredResources
This property indicates the database resources that are registered in the notification registration.

Declaration

```csharp
// C#
public ArrayList RegisteredResources {get;}
```
OracleDependency Properties

Property Value
The registered resources in the notification registration.

Remarks
The ArrayList contains all the command statement or statements that are registered for notification through this OracleDependency object. It is appropriately updated when the Continuous Query Notification is registered by a command execution.

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

RowidInfo
This property specifies whether or not ROWID information is part of change notification events fired whenever data changes on the database.

Declaration

// C#
public OracleRowidInfo RowidInfo {get; set;};

Property Value
An OracleRowidInfo enumeration type that determines the inclusion of ROWID in the change notification event.

Remarks
There are three OracleRowidInfo enumeration types that are valid for this property:
- Default includes ROWID information in the change notification event only if OracleCommand.AddRowid property is set to true or if ROWID is in the select list of the query that is registered for change notification.
- Include includes ROWID information regardless of whether or not ROWID is in the select-list for the query.
- Exclude excludes ROWID information regardless of whether or not ROWID is in the select-list.

For change notification registrations that involve stored procedure executions, change notification events related to the REF CURSOR contain ROWID information only if RowidInfo property is set to OracleRowidInfo.Include.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members
- "OracleRowidInfo Enumeration" on page 9-41

UserName
This property indicates the database user name associated with the OracleDependency instance.
Declaration

// C#
public string UserName{get;}

Property Value
A string that indicates the database user name associated with the OracleDependency instance. This database user registers the Continuous Query Notification request with the database.

Remarks
The UserName property is populated with the user name once the OracleCommand associated with the OracleDependency executes and registers for the notification successfully. Only the database user who creates the notification registration, or the database system administrator, can remove the registration.

The user specified by this property must have the CHANGE NOTIFICATION privilege to register successfully for the Continuous Query Notification with the database.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members
OracleDependency Methods

OracleDependency methods are listed in Table 9–10.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCommandDependency</td>
<td>Binds the OracleDependency instance to the specified OracleCommand instance</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>RemoveRegistration</td>
<td>Removes the specified dependency between the application and the database</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

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

AddCommandDependency

This instance method binds the OracleDependency instance to the specified OracleCommand instance.

Declaration

// C#
Public void AddCommandDependency (OracleCommand cmd);

Parameters

- cmd

  The command that is to be bound to the OracleDependency object.

Exceptions

- ArgumentNullException - The cmd parameter is null.
- InvalidOperationException - The specified OracleCommand instance already contains a notification request.

Remarks

An OracleDependency instance can bind to multiple OracleCommand instances.

While it binds an existing OracleDependency instance to an OracleCommand instance, the AddCommandDependency method creates an OracleNotificationRequest instance, and sets it to the specified OracleCommand.Notification property.

When this method creates an OracleNotificationRequest instance, the following OracleNotificationRequest properties are set:
OracleDependency Class

- IsNotifiedOnce is set to the value True.
- Timeout is set to 50,000 seconds.
- IsPersistent is set to the value False, indicating that the invalidation message is stored in an in-memory queue before delivery.

With this method, multiple commands can be associated with a single Continuous Query Notification registration request. Furthermore, the OracleNotificationRequest attribute values assigned to the OracleCommand can be changed once the association between the OracleCommand and the OracleDependency is established.

However, when multiple OracleCommand objects are associated with a single OracleDependency object, the OracleNotificationRequest attributes (Timeout, IsPersistent, and IsNotifiedOnce) of the first executed OracleCommand object are used for registration, the attributes associated with subsequent OracleCommand executions will be ignored.

Furthermore, once a command associated with an OracleDependency is executed and registered, all other subsequent command executions and registration associated with the same OracleDependency must use a connection with the same "User Id" and "Data Source" connection string attribute value settings. Otherwise, an exception will be thrown.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members
- "OracleDependency(OracleCommand)" on page 9-5 for OracleNotificationRequest property value

RemoveRegistration

This instance method removes the specified dependency between the application and the database. Once the registration of the dependency is removed from the database, the OracleDependency is no longer able to detect any changes that the database undergoes.

Declaration

// C#
public void RemoveRegistration(OracleConnection con)

Parameters

- con
  The connection associated with the OracleDependency instance.

Exceptions

InvalidOperationException - The associated connection is not open.
Remarks

The notification registration associated with the `OracleDependency` instance is removed from the database.

The `OracleConnection` parameter must be in an *opened state*. This instance method does not open the connection implicitly for the application.

An exception is thrown if the dependency is not valid.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleDependency` Class
- `OracleDependency` Members
OracleDependency Class

OracleDependency Events

The OracleDependency event is listed in Table 9–11.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnChange</td>
<td>An event that is sent when a database notification associated with the dependency is received from the database</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members

OnChange

The OnChange event is sent when a database notification associated with the dependency is received from the database. The information related to the notification is stored in the OracleChangeNotificationEventArgs class.

Declaration

// C#
public event OnChangeEventHandler OnChange;

Remarks

The OnChange event occurs if any result set associated with the dependency changes. For objects that are part of a Transaction, notifications will be received for each modified object. This event also occurs for other actions related to database or registration status, such as database shutdowns and startups, or registration timeouts.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleDependency Class
- OracleDependency Members
OracleNotificationRequest Class

An OracleNotificationRequest class represents a notification request to be subscribed in the database. It contains information about the request and the characteristics of the notification. Using the OracleNotificationRequest class, Oracle Data Provider for .NET can create the notification registration in the database.

Class Inheritance
System.Object

Declaration
// C#
public sealed class OracleNotificationRequest

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

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
Comment: Not supported in a .NET stored procedure

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationRequest Members
- OracleNotificationRequest Static Methods
- OracleNotificationRequest Properties
- OracleNotificationRequest Methods
OracleNotificationRequest Members

OracleNotificationRequest members are listed in the following tables.

OracleNotificationRequest Static Method
The OracleNotificationRequest static method is listed in Table 9–12.

Table 9–12 OracleNotificationRequest Static Method

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

OracleNotificationRequest Properties
OracleNotificationRequest properties are listed in Table 9–13.

Table 9–13 OracleNotificationRequest Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotifiedOnce</td>
<td>Indicates whether or not the registration is to be removed upon notification</td>
</tr>
<tr>
<td>IsPersistent</td>
<td>Indicates whether or not the notification message should be queued persistently in the database before delivery</td>
</tr>
<tr>
<td>Timeout</td>
<td>Specifies the time that the registration remains alive</td>
</tr>
<tr>
<td>GroupingNotificationEnabled</td>
<td>Specifies whether grouping notification is enabled or not</td>
</tr>
<tr>
<td>GroupingType</td>
<td>Specifies the type of grouping notification</td>
</tr>
<tr>
<td>GroupingInterval</td>
<td>Specifies the interval between grouping notifications, in seconds</td>
</tr>
</tbody>
</table>

OracleNotificationRequest Methods
OracleNotificationRequest methods are listed in Table 9–14.

Table 9–14 OracleNotificationRequest Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationRequest Members
- OracleNotificationRequest Class
OracleNotificationRequest Static Methods

The OracleNotificationRequest static method is listed in Table 9–15.

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationRequest Members
- OracleNotificationRequest Class
OracleNotificationRequest Properties

The OracleNotificationRequest properties are listed in Table 9–16.

Table 9–16 OracleNotificationRequest Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotifiedOnce</td>
<td>Indicates whether or not the registration is to be removed upon notification</td>
</tr>
<tr>
<td>IsPersistent</td>
<td>Indicates whether or not the notification message should be queued persistently in the database before delivery</td>
</tr>
<tr>
<td>Timeout</td>
<td>Specifies the time that the registration remains alive</td>
</tr>
<tr>
<td>GroupingNotificationEnabled</td>
<td>Specifies whether grouping notification is enabled or not</td>
</tr>
<tr>
<td>GroupingType</td>
<td>Specifies the type of grouping notification</td>
</tr>
<tr>
<td>GroupingInterval</td>
<td>Specifies the interval between grouping notifications, in seconds</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationRequest Members
- OracleNotificationRequest Class

IsNotifiedOnce

This property indicates whether or not the registration is to be removed upon notification.

Declaration

```csharp
// C#
public bool IsNotifiedOnce { get; set; }
```

Property Value

A bool value that indicates whether or not the registration is to be removed upon notification.

Remarks

The default value is false for AQ. This is different from change notification where the default value is true.

Modifying this property after the completion of a successful registration has no effect.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationRequest Members
- OracleNotificationRequest Class
OracleNotificationRequest Properties

IsPersistent

This property indicates whether or not the notification message should be queued persistently in the database until delivery.

**Declaration**

```csharp
// C#
public bool IsPersistent { get; set; }
```

**Property Value**

A `bool` value that indicates whether or not the notifications should be stored persistently in the database until delivery.

When the `IsPersistent` property is set to `true`, the message is queued persistently in the database and cannot be lost upon database failures or shutdowns. When the `IsPersistent` property is set to `false`, the message is stored in an in-memory queue before delivery and could be lost.

This property does not apply to `NotificationRegistration` which is always persistent.

This property only applies to the notification message after it has been sent.

**Remarks**

The default value is `false`.

The database performs faster if the message is stored in an in-memory queue rather than a database queue.

Modifying this property after the completion of a successful registration has no effect.

This property is ignored for grouping notifications.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleNotificationRequest Members`
- `OracleNotificationRequest Class`

Timeout

This property specifies the time, in seconds, that the registration remains alive.

**Declaration**

```csharp
// C#
public long Timeout { get; set; }
```

**Property Value**

A `long` value that specifies the time, in seconds, that the registration remains alive. The valid values for the `Timeout` property are between 0 and 4294967295.

**Exceptions**

`ArgumentOutOfRangeException` - The specified `Timeout` is invalid.
Remarks
The default value is 0 (infinite) for AQ and 50000 for change notification. If the Timeout property is set to 0, then the registration does not expire.

If the registration is removed because the Timeout value has been reached, then the database sends a notification indicating the expiration.

Modifying this property after the completion of a successful registration has no effect.

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

GroupingNotificationEnabled
This property specifies whether grouping notification is enabled or not.

Declaration
// C#
public bool GroupingNotificationEnabled {get; set}

Property Value
A true value indicates that grouping notification is enabled. A false value indicates that grouping notification is disabled.

Remarks
The default value is false.

Modifying this property after the completion of a successful registration has no effect.

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

GroupingType
This property specifies the type of grouping notification.

Declaration
// C#
public OracleAQNotificationGroupingType GroupingType {get; set}

Property Value
An OracleAQNotificationGroupingType enum value

Remarks
The default value is OracleAQNotificationGroupingType.Summary.

Modifying this property after the completion of a successful registration has no effect.
**GroupingInterval**

This property specifies the interval of grouping notification in seconds. The group notifications are delivered at intervals specified by this property.

**Declaration**

```csharp
// C#
public int GroupingInterval {get; set}
```

**Property Value**

An integer specifying the grouping interval in seconds.

**Remarks**

The default value is 600 seconds.

The range of `GroupingInterval` is from 0 to `Int32.MaxValue`. Modifying this property after the completion of a successful registration has no effect.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationRequest Members
- OracleNotificationRequest Class

---

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationRequest Members
- OracleNotificationRequest Class
OracleNotificationRequest Methods

OracleNotificationRequest methods are listed in Table 9–17.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationRequest Members
- OracleNotificationRequest Class
OracleNotificationEventArgs Class

The OracleNotificationEventArgs class provides event data for a notification.

Class Inheritance

System.Object

  System.EventArgs


Declaration

// C#
public sealed class OracleNotificationEventArgs

Thread Safety

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

Requirements

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

Comment: Not supported in a .NET stored procedure

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationEventArgs Members
- OracleNotificationEventArgs Static Fields
- OracleNotificationEventArgs Static Methods
- OracleNotificationEventArgs Properties
- OracleNotificationEventArgs Methods
OracleNotificationEventArgs Members

OracleNotificationEventArgs members are listed in the following tables.

OracleNotificationEventArgs Static Fields
The OracleNotificationEventArgs static field is listed in Table 9–18.

Table 9–18  OracleNotificationEventArgs Static Field

<table>
<thead>
<tr>
<th>Static Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Inherited from System.EventArgs</td>
</tr>
</tbody>
</table>

OracleNotificationEventArgs Static Methods
The OracleNotificationEventArgs static method is listed in Table 9–19.

Table 9–19  OracleNotificationEventArgs Static Method

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

OracleNotificationEventArgs Properties
OracleNotificationEventArgs properties are listed in Table 9–20.

Table 9–20  OracleNotificationEventArgs Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details</td>
<td>Contains detailed information about the current notification</td>
</tr>
<tr>
<td>Info</td>
<td>Indicates the database events for the notification</td>
</tr>
<tr>
<td>ResourceNames</td>
<td>Indicates the database resources related to the current notification</td>
</tr>
<tr>
<td>Source</td>
<td>Returns the database event source for the notification</td>
</tr>
<tr>
<td>Type</td>
<td>Returns the database event type for the notification</td>
</tr>
</tbody>
</table>

OracleNotificationEventArgs Methods
OracleNotificationEventArgs methods are listed in Table 9–21.

Table 9–21  OracleNotificationEventArgs Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationEventArgs Class
OracleNotificationEventArgs Static Fields

The OracleNotificationEventArgs static field is listed in Table 9–22.

<table>
<thead>
<tr>
<th>Static Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Inherited from System.EventArgs</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationEventArgs Class
- OracleNotificationEventArgs Members
OracleNotificationEventArgs Static Methods

The `OracleNotificationEventArgs` static method is listed in Table 9–23.

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleNotificationEventArgs` Class
- `OracleNotificationEventArgs` Members
OracleNotificationEventArgs Properties

OracleNotificationEventArgs properties are listed in Table 9–24.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details</td>
<td>Contains detailed information about the current notification</td>
</tr>
<tr>
<td>Info</td>
<td>Indicates the database events for the notification</td>
</tr>
<tr>
<td>ResourceNames</td>
<td>Indicates the database resources related to the current notification</td>
</tr>
<tr>
<td>Source</td>
<td>Returns the database event source for the notification</td>
</tr>
<tr>
<td>Type</td>
<td>Returns the database event type for the notification</td>
</tr>
</tbody>
</table>

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

Details

This property contains detailed information about the current notification.

Declaration

// C#
Public DataTable Details{get;}

Property Value

A DataTable instance that contains detailed information about the current notification.

Remarks

The returned DataTable object contains column data about the current notification in order as shown in Table 9–25.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourceName</td>
<td>System.String</td>
<td>The resource name of the invalidated object in the format &lt;Schema_name&gt;.&lt;object_name&gt;</td>
</tr>
<tr>
<td>Info</td>
<td>OracleNotificationInfo</td>
<td>The information about the database event that occurs on a resource</td>
</tr>
<tr>
<td>Rowid</td>
<td>System.String</td>
<td>The rowid for the invalidated table row</td>
</tr>
<tr>
<td>QueryId</td>
<td>Int32</td>
<td>The CHANGE_NOTIFICATION_QUERY_ID</td>
</tr>
</tbody>
</table>

The QueryId column contains the CHANGE_NOTIFICATION_QUERY_ID that corresponds to the pseudo-column that may have been retrieved by a SELECT statement at the time of the query-based notification. Also, the OracleDependency
object maintains all the `CHANGE_NOTIFICATION_QUERY_IDS` that are registered with it.

For Continuous Query Notification:

- The `Details` property indicates changes for each invalidated object in the notification in the data table.
- If `ROWID` information is requested, then the `ROWID` information is populated into the `Rowid` column. However, if many rows are modified in a table, then the whole table is invalidated, and `ROWID` information is not provided. Therefore, the `Rowid` column contains all `Null` values.
- If the database event is related to a DDL change of the table or a table drop, then the `Rowid` column is set to `Null`.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleNotificationEventArgs` class
- `OracleNotificationEventArgs Members`

**Info**

This property indicates the database events for the notification.

**Declaration**

```csharp
// C#
public OracleNotificationInfo Info{get;}
```

**Property Value**

An `OracleNotificationInfo` value that indicates the database event for the notification.

**Remarks**

The `OracleNotificationInfo` value is an enumeration type. If several events are received from the invalidation message, the `Info` property is set to one of the `OracleNotificationInfo` enumeration values associated with the database events. For example, if a table has been altered and a new row has been inserted into another table, the `Info` property is set to either `OracleNotificationInfo.Altered` or `OracleNotificationInfo.Insert`.

To obtain more detailed information from the invalidation message, use the `Details` and the `ResourceNames` properties.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleNotificationEventArgs` class
- `OracleNotificationEventArgs Members`
- "Details" on page 9-34
- "ResourceNames" on page 9-36
- "OracleNotificationInfo Enumeration" on page 9-44
OracleNotificationEventArgs Properties

**ResourceNames**
This property indicates the database resources related to the current notification.

**Declaration**

```csharp
// C#
public string[] ResourceNames{get;}
```

**Property Value**
A string array that indicates the database resources related to the current notification.

**Remarks**
For Continuous Query Notification, the `ResourceNames` property contains information about the invalidated object names in the format `<schema_name>.<object_name>`. To obtain more detailed information about the changes for invalidated objects, use the `Details` property.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleNotificationEventArgs Class`
- `OracleNotificationEventArgs Members`
- "Details" on page 9-34

**Source**
This property returns the database event source for the notification.

**Declaration**

```csharp
// C#
public OracleNotificationSource Source{get;}
```

**Property Value**
The `OracleNotificationSource` value for the notification.

**Remarks**
The `OracleNotificationSource` value is an enumeration type. If several event sources are received from the notification message, the `Source` property is set to one of the `OracleNotificationSource` enumeration values related to the database event source. For example, if a table has been altered (by the `ALTER TABLE` command) and a new row has been inserted into the same table, the `Source` property is set to `OracleNotificationSource.Object` or `OracleNotificationSource.Data`.

For Continuous Query Notification:
- When the `Source` property is set to `OracleNotificationSource.Data`:
  - The `Info` property is set to one of the following:
    * `OracleNotificationInfo.Insert`
    * `OracleNotificationInfo.Delete`
    * `OracleNotificationInfo.Update`
The `ResourceNames` property is set, and the elements are set to the invalidated object names.

The `Details` property contains detailed information on the change of each invalidated table.

**When the `Source` property is set to `OracleNotificationSource.Database`:**

- The `Info` property is set to one of the following:
  - `OracleNotificationInfo.Startup`
  - `OracleNotificationInfo.Shutdown`
  - `OracleNotificationInfo.Shutdown_Any`
  - `OracleNotificationInfo.Dropped`

**When the `Source` property is set to `OracleNotificationSource.Object`:**

- The `Info` property is set to either `OracleNotificationInfo.Altered` or `OracleNotificationInfo.Dropped`.
- The `ResourceNames` property is set, and the array elements of the `ResourceNames` property are set to the object names that have been altered or dropped.
- The `Details` property contains detailed information on the changes of the object.

**When the `Source` property is set to `OracleNotificationSource.Subscription`:**

- The `Info` property is set to the following:
  - `OracleNotificationInfo.End`

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleNotificationEventArgs Class`
- `OracleNotificationEventArgs Members`
- "OracleNotificationSource Enumeration" on page 9-43

**Type**

This property returns the database event type for the notification.

**Declaration**

```csharp
// C#
public OracleNotificationType Type{get;}
```

**Property Value**

An `OracleNotificationType` enumeration value that represents the type of the database event notification.

**Remarks**

The `OracleNotificationType` value is an enumeration type. If several event types are received from the notification message, then the `Type` property is set to one of the `OracleNotificationType` enumeration values related to the database event type.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationEventArgs Class
- OracleNotificationEventArgs Members
- "OracleNotificationType Enumeration" on page 9-42
OracleNotificationEventArgs Methods

OracleNotificationEventArgs methods are listed in Table 9–26.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleNotificationEventArgs Class
- OracleNotificationEventArgs Members
OnChangeEventHandler Delegate

The OnChangeEventHandler delegate represents the signature of the method that handles the notification.

Declaration

```csharp
// C#  
public delegate void OnChangeEventHandler(object sender,  
                                          OracleNotificationEventArgs args);
```

Parameters

- `sender`
  The source of the event.
- `args`
  The `OracleNotificationEventArgs` instance that contains the event data.

Requirements

- Assembly: `Oracle.DataAccess.dll`
- ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
- Comment: Not supported in a .NET stored procedure

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleNotificationEventArgs Class`
- `OracleNotificationEventArgs Members`
OracleRowidInfo Enumeration

OracleRowidInfo enumeration values specify whether ROWID information is included as part of the ChangeNotificationEventArgs or not. Table 9–28 lists all the OracleRowidInfo enumeration values with a description of each enumerated value.

Table 9–27  OracleRowidInfo Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>ROWID information is included only if OracleCommand.AddRowid property is set to true or if ROWID column is explicitly included in the query.</td>
</tr>
<tr>
<td>Include</td>
<td>ROWID information is included regardless of whether ROWID is included in the select-list of the query or not.</td>
</tr>
<tr>
<td>Exclude</td>
<td>ROWID information is not included regardless of whether ROWID is included in the select-list of the query or not.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "RowidInfo" on page 9-16
OracleNotificationType Enumeration

OracleNotificationType enumerated values specify the different types that cause the notification.

Table 9–28 lists all the OracleNotificationType enumeration values with a description of each enumerated value.

Table 9–28  OracleNotificationType Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change</td>
<td>A change occurs in the database.</td>
</tr>
<tr>
<td>Subscribe</td>
<td>A change occurs in the subscription.</td>
</tr>
<tr>
<td>Query</td>
<td>A query-based change occurs in the database</td>
</tr>
</tbody>
</table>

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

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

OracleNotificationSource enumerated values specify the different sources that cause notification.

Table 9–29 lists all the OracleNotificationSource enumeration values with a description of each enumerated value.

Table 9–29  OracleNotificationSource Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>The data in a table has changed.</td>
</tr>
<tr>
<td>Database</td>
<td>A database event such as a database startup or shutdown occurs.</td>
</tr>
<tr>
<td>Object</td>
<td>A database object is altered or dropped.</td>
</tr>
<tr>
<td>Subscription</td>
<td>The subscription is changed.</td>
</tr>
</tbody>
</table>

Requirements

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

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

OracleNotificationInfo enumerated values specify the database event that causes the notification.

Table 9–30 lists all the OracleNotificationInfo enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>A row is inserted.</td>
</tr>
<tr>
<td>Delete</td>
<td>A row is deleted.</td>
</tr>
<tr>
<td>Update</td>
<td>A row is updated.</td>
</tr>
<tr>
<td>Startup</td>
<td>A database starts.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>A database shuts down.</td>
</tr>
<tr>
<td>Shutdown_Any</td>
<td>A database instance in a Real Application Cluster (Oracle RAC) environment shuts down.</td>
</tr>
<tr>
<td>Alter</td>
<td>An object is altered.</td>
</tr>
<tr>
<td>Drop</td>
<td>An object or database is dropped.</td>
</tr>
<tr>
<td>End</td>
<td>A registration is removed.</td>
</tr>
<tr>
<td>Error</td>
<td>A notification error occurs.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also: "Oracle.DataAccess.Client Namespace" on page 1-4
This chapter describes the ODP.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

System.Object

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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 10–1.

Table 10–1  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 10–2.

Table 10–2  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 or not 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 10–3.

<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 System.ComponentModel.Component</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleGlobalization Class
- OracleGlobalization Members
OracleGlobalization Static Methods

The OracleGlobalization static methods are listed in Table 10–4.

<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 1-4
- 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 1-4
- 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 1-4
- 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);
// Prints "glob.Language = AMERICAN"
Console.WriteLine('glob.Language = ' + glob.Language);

glob.Dispose();
}
}

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

GetThreadInfo

GetThreadInfo returns or refreshes an OracleGlobalization instance.

Overload List:
- GetThreadInfo()
  This method returns an OracleGlobalization object instance of the current thread.
- GetThreadInfo(OracleGlobalization)
  This method refreshes the OracleGlobalization object instance with the globalization settings of the current thread.

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

GetThreadInfo()

This method returns an OracleGlobalization instance of the current thread.

Declaration

  // C#
  public static OracleGlobalization GetThreadInfo();

Return Value

An OracleGlobalization instance.

Remarks

Initially, GetThreadInfo() returns an OracleGlobalization object that has the same property values as that returned by GetClientInfo(), unless the application changes it by invoking SetThreadInfo().

Example

  // C#

  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 1-4
- OracleGlobalization Class
- OracleGlobalization Members

GetThreadInfo(OracleGlobalization)

This method refreshes the OracleGlobalization object with the globalization settings of the current thread.

Declaration
// C#
public static void GetThreadInfo(OracleGlobalization oraGlob);

Parameters
- oraGlob
  The OracleGlobalization object being updated.

Remarks
Initially GetThreadInfo() returns an OracleGlobalization object that has the same property values as that returned by GetClientInfo(), unless the application changes it by invoking SetThreadInfo().

Example
// C#

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 1-4
- 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);
OracleGlobalization Class

// 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 1-4
■ OracleGlobalization Class
■ OracleGlobalization Members
OracleGlobalization Properties

The OracleGlobalization properties are listed in Table 10–5.

Table 10–5  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 or not 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 1-4
- OracleGlobalization Class
- OracleGlobalization Members

Calendar

This property specifies the calendar system.

Declaration

```csharp
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 or not the OracleGlobalization object represents the settings of the client, thread, or session.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
- 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.
OracleGlobalization Properties

Remarks
The default value is the NLS_COMP setting of the local computer.

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

Currency

This property specifies the string to use as a local currency symbol for the L number format element.

Declaration

```csharp
// 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 1-4
- OracleGlobalization Class
- OracleGlobalization Members
- Oracle Database SQL Reference for further information on the L number format element

DateFormat

This property specifies the date format for Oracle Date type as a string.

Declaration

```csharp
// 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 1-4
- 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**

```csharp
// 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 1-4
- OracleGlobalization Class
- OracleGlobalization Members

**DualCurrency**

This property specifies the dual currency symbol, such as Euro, for the U number format element.

**Declaration**

```csharp
// 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.
**ISOCurrency**

This property specifies the string to use as an international currency symbol for the C number format element.

**Declaration**

```csharp
// C#
public string ISOCurrency {get; set;}
```

**Property Value**

The string used as an international currency symbol.

**Exceptions**

- `ObjectDisposedException` - The object is already disposed.

**Remarks**

The default value is the `NLS_ISO_CURRENCY` setting of the local computer.

**See Also:**

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

```csharp
// 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.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleGlobalization Class
- OracleGlobalization Members
- Oracle Database SQL Reference for further information on the C number format element
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 1-4
- OracleGlobalization Class
- OracleGlobalization Members

### LengthSemantics

This property indicates whether or not 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 1-4
- OracleGlobalization Class
- OracleGlobalization Members

### NCharConversionException

This property determines whether or not 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 1-4
- OracleGlobalization Class
- OracleGlobalization Members
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleGlobalization Class
- OracleGlobalization Members

NumericCharacters
This property specifies the characters used for the decimal character and the group separator character for numeric values in strings.

Declaration
// C#
public string NumericCharacters {get; set;}

Property Value
A string that represents the characters used.

Exceptions
ObjectDisposedException - The object is already disposed.

Remarks
The default value is the NLS_NUMERIC_CHARACTERS setting of the local computer.

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

Sort
This property specifies the collating sequence for ORDER by clause.

Declaration
// C#
public string Sort {get; set;}

Property Value
A string that indicates the collating sequence.

Exceptions
ObjectDisposedException - The object is already disposed.

Remarks
The default value is the NLS_SORT setting of the local computer.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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 1-4
- OracleGlobalization Class
- OracleGlobalization Members
- Oracle Database Globalization Support Guide.

**TimeStampFormat**

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 1-4
- OracleGlobalization Class
- OracleGlobalization Members

**TimeStampTZFormat**

This property specifies the string format for `TimeStampTZ` types.
OracleGlobalization Properties

Declaration

```csharp
// C#
public string TimeStampTZFormat {get; set;}
```

Property Value

The string format for TimeStampTZ types.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the NLS_TIMESTAMP_TZ_FORMAT setting of the local computer.

See Also:

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

TimeZone

This property specifies the time zone region name or hour offset.

Declaration

```csharp
// C#
public string TimeZone {get; set;}
```

Property Value

The string represents the time zone region name or the time zone offset.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is the time zone region name of the local computer. TimeZone is only used when the thread constructs one of the TimeStamp structures. TimeZone has no effect on the session. TimeZone can be either an hour offset, for example, 7:00, or a valid time zone region name that is provided in V$TIMEZONE_NAMES, such as US/Pacific. Time zone abbreviations are not supported.

---

**Note:** PST is a time zone region name as well as a time zone abbreviation; therefore it is accepted by OracleGlobalization.

This property returns an empty string if the OracleGlobalization object is obtained using GetSessionInfo() or GetSessionInfo(OracleGlobalization). Initially, by default, the time zone of the session is identical to the time zone of the thread. Therefore, given that the session time zone is not changed by invoking ALTER SESSION calls, the session time zone can be fetched from the client's globalization settings.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleGlobalization Class
- OracleGlobalization Members
OracleGlobalization Public Methods

OracleGlobalization public methods are listed in Table 10–6.

Table 10–6  OracleGlobalization Public Methods

<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 System.ComponentModel.Component</td>
</tr>
</tbody>
</table>

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

Clone

This method creates a copy of an OracleGlobalization object.

Declaration

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

Return Value
An OracleGlobalization object.

Implements
ICloneable

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleGlobalization Class
- OracleGlobalization Members
This chapter describes the ODP.NET failover classes and enumerations.

This chapter contains these topics:

- OracleFailoverEventArgs Class
- OracleFailoverEventHandler Delegate
- FailoverEvent Enumeration
- FailoverReturnCode Enumeration
- FailoverType Enumeration
OracleFailoverEventArgs Class

The OracleFailoverEventArgs class provides event data for the OracleConnection.Failover event. When database failover occurs, the OracleConnection.Failover event is triggered along with the OracleFailoverEventArgs object that stores the event data.

Class Inheritance

System.Object
  System.EventArgs

Declaration

// C#
public sealed class OracleFailoverEventArgs

Thread Safety

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

Example (Oracle.DataAccess.Client only)

// Transparent Application Failover (TAF) Setup
// Refer Oracle® Database Net Services Administrator's Guide

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

class FailoverSample
{
  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);

    Console.WriteLine("Wait for a failover for 5 seconds");
    Thread.Sleep(5000);

    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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

Comment: Not supported in a .NET stored procedure
OracleFailoverEventArgs Class

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleFailoverEventArgs Members
- OracleFailoverEventArgs Static Methods
- OracleFailoverEventArgs Properties
- OracleFailoverEventArgs Public Methods
- "OracleConnection Class" on page 5-64
- Oracle Net Services Administrator's Guide
OracleFailoverEventArgs Members

OracleFailoverEventArgs members are listed in the following tables.

OracleFailoverEventArgs Static Methods
The OracleFailoverEventArgs static methods are listed in Table 11–1.

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

OracleFailoverEventArgs Properties
The OracleFailoverEventArgs properties are listed in Table 11–2.

<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 11–3.

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleFailoverEventArgs Class
- "FailoverType Enumeration" on page 11-12
OracleFailoverEventArgs Static Methods

The OracleFailoverEventArgs static methods are listed in Table 11-1.

Table 11-4 OracleFailoverEventArgs Static Methods

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

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members
OracleFailoverEventArgs Properties

The `OracleFailoverEventArgs` properties are listed in Table 11–5.

<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 1-4
- 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 1-4
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members
- "FailoverType Enumeration" on page 11-12

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 1-4
- OracleFailoverEventArgs Class
- OracleFailoverEventArgs Members
- "FailoverEvent Enumeration" on page 11-10
OracleFailoverEventArgs Public Methods

The OracleFailoverEventArgs public methods are listed in Table 11–6.

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- 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)
```

**Requirements**

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

Comment: Not supported in a .NET stored procedure

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleFailoverEventArgs Class`
- `OracleFailoverEventArgs Members`
- "Failover" on page 5-112
FailoverEvent Enumeration

FailoverEvent enumerated values are used to specify the state of the failover. Table 11–7 lists all the FailoverEvent enumeration values with a description of each enumerated value.

Table 11–7 FailoverEvent Enumeration Values

<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>

Requirements

Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- FailoverEvent Enumeration on page 11-10
- "OracleFailoverEventArgs Class" on page 11-2
- "FailoverEvent" on page 11-7
- Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide
- 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 11–8 lists the FailoverReturnCode enumeration values with a description of each enumerated value.

Table 11–8  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>

Requirements

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- FailoverEvent Enumeration on page 11-10
- "OracleFailoverEventArgs Class” on page 11-2
- "FailoverEvent” on page 11-7
- Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide
- Oracle Net Services Reference Guide
FailoverType Enumeration

FailoverType enumerated values are used to indicate the type of failover event that was raised.

Table 11–9 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>

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- FailoverEvent Enumeration on page 11-10
- "OracleFailoverEventArgs Class" on page 11-2
- "FailoverType" on page 11-7
- Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide
- Oracle Net Services Reference Guide
This chapter describes the following Oracle Data Provider for .NET classes:

- OracleAQAgent Class
- OracleAQDequeueOptions Class
- OracleAQEnqueueOptions Class
- OracleAQMessage Class
- OracleAQMessageAvailableEventArgs Class
- OracleAQMessageAvailableEventHandler Delegate
- OracleAQQueue Class
- OracleAQDequeueMode Enumeration
- OracleAQMessageDeliveryMode Enumeration
- OracleAQMessageState Enumeration
- OracleAQMessageType Enumeration
- OracleAQNavigationMode Enumeration
- OracleAQNotificationGroupingType Enumeration
- OracleAQNotificationType Enumeration
- OracleAQVisibilityMode Enumeration
OracleAQAgent Class

The `OracleAQAgent` class represents agents that may be senders or recipients of a message.

**Class Inheritance**

```
System.Object
   OracleAQAgent
```

**Declaration**

```
// C#
public sealed class OracleAQAgent
```

**Thread Safety**

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

**Remarks**

An agent may be a consumer, another queue, or a consumer of another queue. The queue may be either local or remote. A remote queue is specified through a database link.

**Requirements**

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQAgent Members
- OracleAQAgent Constructors
- OracleAQAgent Properties
OracleAQAgent Members

OracleAQAgent members are listed in the following tables.

OracleAQAgent Constructors
OracleAQAgent constructors are listed in Table 12–1.

Table 12–1 OracleAQAgent Constructors

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

OracleAQAgent Properties
OracleAQAgent properties are listed in Table 12–2.

Table 12–2 OracleAQAgent Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Specifies the address of the agent.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies the name of the agent.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQAgent Class
OracleAQAgent Constructors

OracleAQAgent constructors instantiate new instances of the OracleAQAgent class.

Overload List:
- OracleAQAgent (string)
  This constructor instantiates the OracleAQAgent class using the specified name.
- OracleAQAgent (string, string)
  This constructor instantiates the OracleAQAgent class using the specified name and address.

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

OracleAQAgent (string)
This constructor instantiates the OracleAQAgent class using the specified name.

Declaration
// C#
public OracleAgent(string name);

Parameters
- name
  The name of the agent.

Exceptions
ArgumentNullException - The name parameter is null.
ArgumentException - The name parameter is empty.

Remarks
The agent name signifies the name of a producer or consumer of a message. In the context of functionality exposed by Listen, an agent name corresponds to the name of a consumer for which a message is expected on a multiconsumer queue. It may also be set on a message to signify sender identification or intended recipients of the message.

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

OracleAQAgent (string, string)
This constructor instantiates the OracleAQAgent class using the specified name and address.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQAgent Class
- OracleAQAgent Members
Declaration

```csharp
// C#
public OracleAQAgent(string name, string address);
```

Parameters

- **name**
  
  The name of the agent.

- **address**
  
  The address is of the form `[schema.]queue[dblink].`

Exceptions

- ArgumentNullException - The `address` parameter is null.
- ArgumentException - The `address` parameter is empty.

Remarks

The agent name signifies the name of a producer or consumer of a message. In the context of functionality exposed by `Listen`, an agent name corresponds to the name of a consumer for which a message is expected on a multiconsumer queue.

The `name` parameter can be specified as `null` in this constructor. In such a scenario, the agent only has an `address`.

The `address` parameter signifies the name of the queue against which this agent listens for new messages. The `address` represents a queue at a local or remote database. The validity of the `address` is not checked implicitly. The exceptions due to wrong `address` are thrown only during database operations such as `Listen`.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQAgent Class
- OracleAQAgent Members
OracleAQAgent Properties

OracleAQAgent properties are listed in Table 12–3.

Table 12–3 OracleAQAgent Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Specifies the address of the agent.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies the name of the agent.</td>
</tr>
</tbody>
</table>

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

Address

This instance property specifies the address of the agent.

Declaration

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

Property Value

A string that specifies the agent address.

Remarks

The address represents a queue at a local or remote database. The default value is null. The address of the agent is of the form `[schema.][queue[@dblink]]`. The string length can be up to 128 characters.

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

Name

This instance property specifies the name of the agent.

Declaration

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

Property Value

A string.
Remarks
The default is null. The string length can be up to 30 characters. A non-null value implies that this agent name either corresponds to a consumer name in a multiconsumer queue, or a recipient as specified in message properties.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQAgent Class
- OracleAQAgent Members
OracleAQDequeueOptions Class

An OracleAQDequeueOptions object represents the options available when dequeuing a message from an OracleAQQueue object.

Class Inheritance
System.Object
   OracleAQDequeueOptions

Declaration
// C#
public sealed class OracleAQDequeueOptions : ICloneable

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

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Members
- OracleAQDequeueOptions Constructor
- OracleAQDequeueOptions Properties
- OracleAQDequeueOptions Public Methods
OracleAQDequeueOptions Members

OracleAQDequeueOptions members are listed in the following tables.

OracleAQDequeueOptions Constructor
The OracleAQDequeueOptions constructor is listed in Table 12–4.

Table 12–4 OracleAQDequeueOptions Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleAQDequeueOptions</td>
<td>Instantiates a new instance of the OracleAQDequeueOptions class</td>
</tr>
<tr>
<td>Constructor</td>
<td></td>
</tr>
</tbody>
</table>

OracleAQDequeueOptions Properties
OracleAQDequeueOptions properties are listed in Table 12–5.

Table 12–5 OracleAQDequeueOptions Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsumerName</td>
<td>Specifies the consumer name for which to dequeue the message</td>
</tr>
<tr>
<td>Correlation</td>
<td>Specifies the correlation identifier of the message to be dequeued</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Specifies the expected delivery mode of the message being dequeued</td>
</tr>
<tr>
<td>DequeueMode</td>
<td>Specifies the locking behavior associated with the dequeue operation</td>
</tr>
<tr>
<td>MessageId</td>
<td>Specifies the message identifier of the message to be dequeued</td>
</tr>
<tr>
<td>NavigationMode</td>
<td>Specifies the position of the message that will be retrieved</td>
</tr>
<tr>
<td>ProviderSpecificType</td>
<td>Specifies whether the payload of a dequeued message is provided as an ODP.NET specific type or a .NET type</td>
</tr>
<tr>
<td>Visibility</td>
<td>Specifies whether or not the new message is dequeued as part of the current transaction</td>
</tr>
<tr>
<td>Wait</td>
<td>Specifies the wait time, in seconds, for a message that matches the search criteria</td>
</tr>
</tbody>
</table>

OracleAQDequeueOptions Public Methods
OracleAQDequeueOptions public methods are listed in Table 12–6.

Table 12–6 OracleAQDequeueOptions Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleAQDequeueOptions object.</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Class
OracleAQDequeueOptions Constructor

The OracleAQDequeueOptions constructor creates an instance of the OracleAQDequeueOptions class and sets all its properties to their default values.

Declaration

// C#
public OracleAQDequeueOptions();

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Class
- OracleAQDequeueOptions Members
OracleAQDequeueOptions Properties

OracleAQDequeueOptions properties are listed in Table 12–7.

Table 12–7  OracleAQDequeueOptions Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsumerName</td>
<td>Specifies the consumer name for which to dequeue the message</td>
</tr>
<tr>
<td>Correlation</td>
<td>Specifies the correlation identifier of the message to be dequeued</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Specifies the expected delivery mode of the message being dequeued</td>
</tr>
<tr>
<td>DequeueMode</td>
<td>Specifies the locking behavior associated with the dequeue operation</td>
</tr>
<tr>
<td>MessageId</td>
<td>Specifies the message identifier of the message to be dequeued</td>
</tr>
<tr>
<td>NavigationMode</td>
<td>Specifies the position of the message that will be retrieved</td>
</tr>
<tr>
<td>ProviderSpecificType</td>
<td>Specifies whether the payload of a dequeued message is provided as an ODP.NET specific type or a .NET type</td>
</tr>
<tr>
<td>Visibility</td>
<td>Specifies whether or not the new message is dequeued as part of the current transaction</td>
</tr>
<tr>
<td>Wait</td>
<td>Specifies the wait time, in seconds, for a message that matches the search criteria</td>
</tr>
</tbody>
</table>

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

ConsumerName

This instance property specifies the consumer name for which to dequeue the message.

Declaration

// C#
public string ConsumerName {get;set;}

Property Value

A string.

Remarks

The ConsumerName property only accesses those messages that match the consumer name. If a queue is not set up for multiple consumers, then this field should be set to null.
Correlation

This instance property specifies the correlation identifier of the message to be dequeued.

Declaration

// C#
public string Correlation {get;set;}

Property Value

A string.

Remarks

This property specifies the identification of the message to be dequeued. Special pattern matching characters, such as the percent sign (%) and the underscore (_) can be used. If more than one message satisfies the pattern, then the order of dequeuing is undetermined.

The maximum length of Correlation is 128.

MessageId and Correlation are two independent identifiers. While MessageId is unique for a message, a group of messages can be assigned the same Correlation. Also, pattern matching is possible only with Correlation.

See Also:

■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleAQDequeueOptions Class
■ OracleAQDequeueOptions Members

DeliveryMode

This instance property specifies the expected delivery mode of the message being dequeued.

Declaration

// C#
public OracleAQMessageDeliveryMode DeliveryMode {get;set;}

Property Value

An OracleAQMessageDeliveryMode enumerated value.

Remarks

This property specifies the type of messages to be dequeued. It can be set to dequeue either persistent or buffered messages, or both from a queue. The following values are valid:

■ OracleAQMessageDeliveryMode.Persistent
OracleAQDequeueOptions Properties

- **OracleAQMessageDeliveryMode.Buffered**
- **OracleAQMessageDeliveryMode.PersistentOrBuffered**

The default value is **OracleAQMessageDeliveryMode.Persistent**. Buffered messaging is supported in all queue tables created with a database compatibility level of 8.1 or higher.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Class
- OracleAQDequeueOptions Members

**DequeueMode**

This instance property specifies the locking behavior associated with the dequeue operation.

**Declaration**

```csharp
// C#
public OracleAQDequeueMode DequeueMode {get;set;}
```

**Property Value**

An **OracleAQDequeueMode** enumerated value.

**Exceptions**

`ArgumentNullException` - The specified DequeueMode value is invalid.

**Remarks**

The default value is **OracleAQDequeueMode.Remove**.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Class
- OracleAQDequeueOptions Members

**MessageId**

This instance property specifies the message identifier of the message to be dequeued.

**Declaration**

```csharp
// C#
public byte[] MessageId {get;set;}
```

**Property Value**

A `byte[]`.

**Remarks**

The dequeue operation succeeds only if the message ID of the message being dequeued matches with the message ID specified.

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

NavigationMode
This instance property specifies the position of the message that will be retrieved.

Declaration

```
// C#
public OracleAQNavigationMode NavigationMode {get;set;}
```

Property Value
An OracleAQNavigationMode enumerated value.

Exceptions
ArgumentOutOfRangeException - The specified NavigationMode value is invalid.

Remarks
The default value is OracleAQNavigationMode.NextMessage.

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

ProviderSpecificType
This property specifies whether the payload of a dequeued message is provided as an ODP.NET specific type or a .NET type.

Declaration

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

Property Value
A bool.

Remarks
The default value of this property is false. For a discussion of how this property affects payload type, refer to "MessageType" under the OracleAQQueue class.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Class
- OracleAQDequeueOptions Members
- "MessageType" on page 12-60
Visibility

This instance property specifies whether or not the new message is dequeued as part of the current transaction.

Declaration

// C#
public OracleAQVisibilityMode Visibility {get;set;}

Property Value

An OracleAQVisibilityMode enumerated value.

Exceptions

ArgumentOutOfRangeException - The Visibility value specified is invalid.

Remarks

The default value is OracleAQVisibilityMode.OnCommit. You must use transactions when using the default value for this property. This ensures that applications do not lose messages and the messages are appropriately removed from the queue after the dequeue operation is successful. If transactions are not used when using the default visibility mode of OracleAQVisibilityMode.OnCommit, then messages are not removed from the queue.

Using the alternative visibility mode value, OracleAQVisibilityMode.Immediate can eliminate the need to create, commit, and rollback a transaction. However, if an error occurs during the dequeue operation, then the message may be lost.

The visibility parameter is ignored when DequeueMode is set to OracleAQDequeueMode.Browse.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Class
- OracleAQDequeueOptions Members

Wait

This instance property specifies the wait time, in seconds, for a message that matches the search criteria.

Declaration

// C#
public int Wait {get;set;}

Property Value

Any positive integer value or 0 or -1.

Exceptions

ArgumentOutOfRangeException - The specified Wait value is invalid.

Remarks

The default value is -1, which implies an infinite wait. The following values are valid:
■ Positive integer: Wait time in seconds.
■ -1: Wait forever.
■ 0: Do not wait.

A value of less than -1 raises an ArgumentOutOfRangeException.

This parameter is ignored if messages in the same group are being dequeued.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Class
- OracleAQDequeueOptions Members
OracleAQDequeueOptions Public Methods

The OracleAQDequeueOptions public method is listed in Table 12–8.

Table 12–8 OracleAQDequeueOptions Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleAQDequeueOptions object</td>
</tr>
</tbody>
</table>

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

Clone

This method creates a copy of an OracleAQDequeueOptions object.

Declaration
// C#
public object Clone();

Return Value
An OracleAQDequeueOptions object.

Implements
ICloneable.

Remarks
The cloned object has the same property values as the object being cloned.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQDequeueOptions Class
- OracleAQDequeueOptions Members
OracleAQEnqueueOptions Class

The OracleAQEnqueueOptions class represents the options available when enqueuing a message to an OracleAQQueue.

Class Inheritance
System.Object
   OracleAQEnqueueOptions

Declaration
// C#
public sealed class OracleAQEnqueueOptions : ICloneable

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

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQEnqueueOptions Members
- OracleAQEnqueueOptions Constructor
- OracleAQEnqueueOptions Properties
- OracleAQEnqueueOptions Public Methods
OracleAQEnqueueOptions Members

The OracleAQEnqueueOptions members are listed in the following tables.

OracleAQEnqueueOptions Constructor

OracleAQEnqueueOptions constructor is listed in Table 12–9.

Table 12–9  OracleAQEnqueueOptions Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleAQEnqueueOptions Constructor</td>
<td>Instantiates a new instance of the OracleAQEnqueueOptions class.</td>
</tr>
</tbody>
</table>

OracleAQEnqueueOptions Properties

OracleAQEnqueueOptions properties are listed in Table 12–10.

Table 12–10  OracleAQEnqueueOptions Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeliveryMode</td>
<td>Specifies the delivery mode of the message being enqueued.</td>
</tr>
<tr>
<td>Visibility</td>
<td>Specifies whether or not the new message is enqueued as part of the current transaction.</td>
</tr>
</tbody>
</table>

OracleAQEnqueueOptions Public Methods

The OracleAQEnqueueOptions public method is listed in Table 12–11.

Table 12–11  OracleAQEnqueueOptions Public Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleAQEnqueueOptions object.</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQEnqueueOptions Class
OracleAQEnqueueOptions Constructor

This constructor creates an instance of the OracleAQEnqueueOptions class with default property values.

Declaration

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQEnqueueOptions Class
- OracleAQEnqueueOptions Members
OracleAQEnqueueOptions Properties

OracleAQEnqueueOptions properties are listed in Table 12–12.

Table 12–12 OracleAQEnqueueOptions Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeliveryMode</td>
<td>Specifies the delivery mode of the message being enqueued.</td>
</tr>
<tr>
<td>Visibility</td>
<td>Specifies whether or not the new message is enqueued as part of the current transaction.</td>
</tr>
</tbody>
</table>

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

DeliveryMode

This instance property specifies the delivery mode of the message being enqueued.

Declaration

// C#
public OracleAQMessageDeliveryMode DeliveryMode {get;set;}

Exceptions

ArgumentOutOfRangeException - The specified Visibility value is invalid.

Remarks

The valid values can be any of the following enumerated values:
- OracleAQMessageDeliveryMode.Persistent
- OracleAQMessageDeliveryMode Buffered

The default is OracleAQMessageDeliveryMode.Persistent.

OracleAQMessageDeliveryMode Buffered cannot be set on this property.

OracleAQMessageDeliveryMode Buffered can be specified only with Oracle Database 10g release 2 (10.2) or higher. Buffered messaging is supported in all queue tables created with a database compatibility level of 8.1 or higher.

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

Visibility

This instance property specifies whether or not the new message is enqueued as part of the current transaction.
Declaration

// C#
public OracleAQVisibilityMode Visibility {get;set;}

Property Value
An OracleAQVisibilityMode enumerated value.

Exceptions
ArgumentOutOfRangeException - The specified Visibility value is invalid.

Remarks
The default value is OracleAQVisibilityMode.OnCommit. You must use transactions when using the default value. If transactions are not used when using the default visibility mode of OracleAQVisibilityMode.OnCommit, then messages are not enqueued to the queue.

Using the alternative visibility mode value, OracleAQVisibilityMode.Immediate eliminates the need to use a transaction. The queue is not affected in case the enqueue operation fails. The message does not get enqueued to the queue for such cases.

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleAQEnqueueOptions Class
- OracleAQEnqueueOptions Members
OracleAQEnqueueOptions Public Methods

OracleAQEnqueueOptions public method is listed in Table 12–13.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a copy of an OracleAQEnqueueOptions object.</td>
</tr>
</tbody>
</table>

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

Clone

This method creates a copy of an OracleAQEnqueueOptions object.

Declaration

// C#
public object Clone();

Return Value
An OracleAQEnqueueOptions 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 1-4
- OracleAQEnqueueOptions Class
- OracleAQEnqueueOptions Members
OracleAQMessage Class

An OracleAQMessage object represents a message to be enqueued and dequeued.

Class Inheritance
System.Object
    OracleAQMessage

Declaration
// C#
public sealed class OracleAQMessage

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

Remarks
An OracleAQMessage object consists of control information (metadata) and Payload (data). The control information is exposed by various properties on the OracleAQMessage object and is used by Oracle Streams Advanced Queuing to manage messages. The payload is the information stored in the queue.

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessage Members
- OracleAQMessage Constructors
- OracleAQMessage Properties
OracleAQMessage Members

OracleAQMessage members are listed in the following tables.

OracleAQMessage Constructor

OracleAQMessage constructors are listed in Table 12–14.

<table>
<thead>
<tr>
<th>Table 12–14 OracleAQMessage Constructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor</td>
</tr>
<tr>
<td>OracleAQMessage Constructors</td>
</tr>
</tbody>
</table>

OracleAQMessage Properties

OracleAQMessage properties are listed in Table 12–15.

<table>
<thead>
<tr>
<th>Table 12–15 OracleAQMessage Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
</tr>
<tr>
<td>Correlation</td>
</tr>
<tr>
<td>Delay</td>
</tr>
<tr>
<td>DeliveryMode</td>
</tr>
<tr>
<td>DequeueAttempts</td>
</tr>
<tr>
<td>EnqueueTime</td>
</tr>
<tr>
<td>ExceptionQueue</td>
</tr>
<tr>
<td>Expiration</td>
</tr>
<tr>
<td>MessageId</td>
</tr>
<tr>
<td>OriginalMessageId</td>
</tr>
<tr>
<td>Payload</td>
</tr>
<tr>
<td>Priority</td>
</tr>
<tr>
<td>Recipients</td>
</tr>
<tr>
<td>SenderId</td>
</tr>
<tr>
<td>State</td>
</tr>
<tr>
<td>TransactionGroup</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessage Class
OracleAQMessage Constructors

OracleAQMessage constructors create new instances of the OracleAQMessage class.

Overload List:

- **OracleAQMessage()**
  
  This constructor instantiates the OracleAQMessage class.

- **OracleAQMessage(Object)**
  
  This constructor instantiates the OracleAQMessage class using the object provided as the payload.

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

OracleAQMessage()

This constructor instantiates the OracleAQMessage class.

Declaration

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

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessage Class
- OracleAQMessage Members

OracleAQMessage(Object)

This constructor instantiates the OracleAQMessage class using the Object provided as the payload.

Declaration

```csharp
// C#
public OracleAQMessage(Object payload);
```

Parameters

- **payload**

  An Object specifying payload. It can be one of the following types:
  - byte[]
  - IOracleCustomType
  - OracleBinary
  - OracleXmlType
  - String
- XmlReader

Exceptions

ArgumentException - The specified payload value is of invalid type.

Remarks

The ODP.NET AQ implementation currently does not support user defined types with LOB attributes. It also does not support other variants of user defined types such as VARRAY and nested tables, as Oracle Streams AQ does not support them inherently.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessage Class
- OracleAQMessage Members
OracleAQMessage Properties

OracleAQMessage properties are listed in Table 12–16.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation</td>
<td>Specifies an identification for the message.</td>
</tr>
<tr>
<td>Delay</td>
<td>Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Specifies the delivery mode of the dequeued message.</td>
</tr>
<tr>
<td>DequeueAttempts</td>
<td>Returns the number of attempts that have been made to dequeue the message.</td>
</tr>
<tr>
<td>EnqueueTime</td>
<td>Specifies the time when the message was enqueued.</td>
</tr>
<tr>
<td>ExceptionQueue</td>
<td>Specifies the name of the queue that the message should be moved to if it cannot be processed successfully.</td>
</tr>
<tr>
<td>Expiration</td>
<td>Specifies the duration, in seconds, for which an enqueued message is available for dequeuing.</td>
</tr>
<tr>
<td>MessageId</td>
<td>Returns the message identifier.</td>
</tr>
<tr>
<td>OriginalMessageId</td>
<td>Specifies the identifier of the message in the last queue that generated this message.</td>
</tr>
<tr>
<td>Payload</td>
<td>Specifies the payload of the message.</td>
</tr>
<tr>
<td>Priority</td>
<td>Specifies the priority of the message.</td>
</tr>
<tr>
<td>Recipients</td>
<td>Specifies the list of recipients that overrides the default queue subscribers.</td>
</tr>
<tr>
<td>SenderId</td>
<td>Identifies the original sender of the message.</td>
</tr>
<tr>
<td>State</td>
<td>Specifies the state of the message at the time of dequeue.</td>
</tr>
<tr>
<td>TransactionGroup</td>
<td>Specifies the transaction group for the dequeued message.</td>
</tr>
</tbody>
</table>

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

Correlation

This instance property specifies an identification for the message.

Declaration

// C#
public string Correlation {get;set;}
Property Value

A string that specifies the identification for the message.

Remarks

The producer of a message can set this property at the time of enqueuing. The consumer can then use this identification to dequeue specific messages by setting the Correlation property of an OracleAQDequeueOptions object. For more information regarding dequeuing messages based on Correlation, refer to "Correlation" under the OracleAQDequeueOptions class.

See Also:

■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleAQMessage Class
■ OracleAQMessage Members
■ "Correlation" on page 12-13

Delay

This instance property specifies the duration, in seconds, after which an enqueued message is available for dequeuing.

Declaration

// C#
public int Delay {get;set;}

Property Value

An integer that indicates the number of seconds after which an enqueued message is available for dequeuing.

Exceptions

ArgumentException - The value specified is less than 0.

Remarks

This property delays the immediate consumption of an enqueued message. The following are valid values for this property:

■ Positive integer - Indicates the delay in seconds.
■ 0 - indicates that the message is immediately available for dequeuing.

The default value is 0. The Delay property is not supported with buffered messaging.

See Also:

■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleAQMessage Class
■ OracleAQMessage Members

DeliveryMode

This instance property specifies the delivery mode of the dequeued message.
OracleAQMessage Properties

Declaration

// C#
public OracleAQMessageDeliveryMode DeliveryMode {get;}

Property Value
An OracleAQMessageDeliveryMode enumerated value (OracleAQMessageDeliveryMode.Persistent or OracleAQMessageDeliveryMode.Buffered).

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

DequeueAttempts
This instance property returns the number of attempts that have been made to dequeue the message.

Declaration

// C#
public int DequeueAttempts {get;}

Property Value
An integer that indicates the number of dequeue attempts.

Remarks
This property is available in an OracleAQMessage after the message has been dequeued from a queue.

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

EnqueueTime
This instance property specifies the time when the message was enqueued.

Declaration

// C#
public DateTime EnqueueTime {get;}

Property Value
A DateTime object.

Remarks
This property is available after the message is dequeued. It provides the enqueue time of a dequeued message.
ExceptionQueue

This instance property specifies the name of the queue that the message should be moved to if it cannot be processed successfully.

Declaration

```csharp
// C#
public string ExceptionQueue {get;set;}
```

Property Value

The name of the queue that a message should be moved to if it cannot be processed successfully. The default value is `null`.

Remarks

This property specifies the queue that a message should be moved to if the message has expired or if the number of unsuccessful dequeue attempts have exceeded the `max_retries` value for the queue.

If this property is not set or the specified exception queue name does not exist, then the default exception queue associated with the queue table is used.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessage Class
- OracleAQMessage Members

Expiration

This instance property specifies the duration, in seconds, for which an enqueued message is available for dequeuing.

Declaration

```csharp
// C#
pubic int Expiration {get;set;}
```

Property Value

An integer that specifies the number of seconds an enqueued message is available for dequeuing.

Exceptions

- ArgumentException: The value specified is less than -1.

Remarks

The value specified is an offset from the value specified in the `Delay` property.

The following are valid values for the property:

- Positive integer - Indicates the expiration in seconds.
-1 - Indicates that the message never expires.

The default value is -1. When a message expires, the message moves from the READY state to the EXPIRED state.

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

### MessageId

This instance property returns the message identifier.

**Declaration**

```csharp
// C#
public byte[] MessageId {get;}
```

**Property Value**

A `byte[]` that specifies the message identifier.

**Remarks**

This property is available after an enqueue or dequeue operation. Dequeued buffered messages have a null value for `MessageId`.

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

### OriginalMessageId

This instance property specifies the identifier of the message in the last queue that generated this message.

**Declaration**

```csharp
// C#
public byte[] OriginalMessageId {get;}
```

**Property Value**

A `byte[]` that specifies the original message identifier.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessage Class
- OracleAQMessage Members

### Payload

This instance property specifies the payload of the message.
Declaration

// C#
public Object Payload {get;set;}

Property Value
An Object that specifies the payload of the message.

Exceptions
ArgumentException - The specified object is not one of the allowed types.

Remarks
For a complete discussion of various payload types, refer to "MessageType" under the OracleAQQueue class.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessage Class
- OracleAQMessage Members
- "MessageType" on page 12-60

Priority
This instance property specifies the priority of the message.

Declaration

// C#
public int Priority {get;set;}

Property Value
An integer that specifies the priority of the message.

Remarks
The default value is 0. In order to take effect, this property must be set prior to enqueuing the message.
Smaller values indicate higher priority for the message. Negative values may also be used.
The priority of an enqueued message is useful for priority-based dequeuing.

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

Recipients
This instance property specifies the list of recipients that overrides the default queue subscribers.

Declaration

// C#
public OracleAQAgent[] Recipients {get; set}

**Property Value**
An OracleAQAgent[].

**Remarks**
This recipient list is valid only for messages being enqueued to multiconsumer queues. The list of recipients is not returned with the message at the time of dequeuing.

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

**SenderId**

This instance property identifies the original sender of the message.

**Declaration**
// C#
public OracleAQAgent SenderId {get; set}

**Property Value**
An OracleAQAgent object.

**Remarks**
Sender identification is supported in all queue tables created with a database compatibility level of 8.1 or higher.

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

**State**

This instance property specifies the state of the message at the time of dequeue.

**Declaration**
// C#
public OracleAQMessageState State {get;}

**Property Value**
An OracleAQMessageState enumerated value.

**Remarks**
This property is available after the message is dequeued. The state of buffered messages dequeued by specifying Correlation under dequeue options is always OracleAQMessageState.Ready.
TransactionGroup

This instance property specifies the transaction group for the dequeued message.

Declaration

// C#
public string TransactionGroup {get;}

Property Value

A string that specifies the transaction group.

Remarks

This property is set only after the call to DequeueArray. This property is supported only when using Oracle Database 10g database or higher.

Messages belonging to one queue can be grouped to form a set that can only be consumed by one user at a time. This requires that the queue be created in a queue table that is enabled for message grouping. All messages belonging to a group must be created in the same transaction. Also, all messages created in one transaction belong to the same group.

See Also:

■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleAQMessage Class
■ OracleAQMessage Members
OracleAQMessageAvailableEventArgs Class

The OracleAQMessageAvailableEventArgs class provides event data for the OracleAQQueue.MessageAvailable event.

Class Inheritance
System.Object
    System.EventArgs

Declaration
    // C#
    public sealed class OracleAQMessageAvailableEventArgs

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

Remarks
This class cannot be inherited.

For detailed information on all the inherited properties and methods, please read the documentation provided by Microsoft's .NET Documentation.

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessageAvailableEventArgs Members
- OracleAQMessageAvailableEventArgs Properties
OracleAQMessageAvailableEventArgs Members

OracleAQMessageAvailableEventArgs members are listed in the following tables.

OracleAQMessageAvailableEventArgs Properties
OracleAQMessageAvailableEventArgs properties are listed in Table 12–17.

Table 12–17 OracleAQMessageAvailableEventArgs Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvailableMessages</td>
<td>Specifies the number of messages that raised this notification.</td>
</tr>
<tr>
<td>ConsumerName</td>
<td>Provides the name of the consumer for which the message is available for dequeuing.</td>
</tr>
<tr>
<td>Correlation</td>
<td>Provides the name of the consumer for which the message is available for dequeuing.</td>
</tr>
<tr>
<td>Delay</td>
<td>Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Specifies the delivery mode of the message.</td>
</tr>
<tr>
<td>EnqueueTime</td>
<td>Specifies the time when the message was enqueued.</td>
</tr>
<tr>
<td>ExceptionQueue</td>
<td>Specifies the name of the queue that the message is moved to if it cannot be processed successfully.</td>
</tr>
<tr>
<td>Expiration</td>
<td>Specifies the duration, in seconds, for which an enqueued message is available for dequeuing before expiring.</td>
</tr>
<tr>
<td>MessageId</td>
<td>Returns an array of message identifiers.</td>
</tr>
<tr>
<td>NotificationType</td>
<td>Indicates the type of notification such as regular, grouping, or timeout.</td>
</tr>
<tr>
<td>OriginalMessageId</td>
<td>Specifies the ID of the message, in the last queue, that generated this message.</td>
</tr>
<tr>
<td>Priority</td>
<td>Specifies the priority of the message.</td>
</tr>
<tr>
<td>QueueName</td>
<td>Indicates the name of the queue that contains the message to be dequeued.</td>
</tr>
<tr>
<td>SenderId</td>
<td>Identifies the original sender of the message.</td>
</tr>
<tr>
<td>State</td>
<td>Specifies the state of the message.</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleAQMessageAvailableEventArgs Class
OracleAQMessageAvailableEventArgs Properties

OracleAQMessageAvailableEventArgs properties are listed in Table 12–18.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvailableMessages</td>
<td>Specifies the number of messages that raised this notification.</td>
</tr>
<tr>
<td>ConsumerName</td>
<td>Provides the name of the consumer for which the message is available for dequeuing.</td>
</tr>
<tr>
<td>Correlation</td>
<td>Provides the name of the consumer for which the message is available for dequeuing.</td>
</tr>
<tr>
<td>Delay</td>
<td>Specifies the duration, in seconds, after which an enqueued message is available for dequeuing.</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Specifies the delivery mode of the message.</td>
</tr>
<tr>
<td>EnqueueTime</td>
<td>Specifies the time when the message was enqueued.</td>
</tr>
<tr>
<td>ExceptionQueue</td>
<td>Specifies the name of the queue that the message is moved to if it cannot be processed successfully.</td>
</tr>
<tr>
<td>Expiration</td>
<td>Specifies the duration, in seconds, for which an enqueued message is available for dequeuing before expiring.</td>
</tr>
<tr>
<td>MessageId</td>
<td>Returns an array of message identifiers.</td>
</tr>
<tr>
<td>NotificationType</td>
<td>Indicates the type of notification such as regular, grouping, or timeout.</td>
</tr>
<tr>
<td>OriginalMessageId</td>
<td>Specifies the ID of the message, in the last queue, that generated this message.</td>
</tr>
<tr>
<td>Priority</td>
<td>Specifies the priority of the message.</td>
</tr>
<tr>
<td>QueueName</td>
<td>Indicates the name of the queue that contains the message to be dequeued.</td>
</tr>
<tr>
<td>SenderId</td>
<td>Identifies the original sender of the message.</td>
</tr>
<tr>
<td>State</td>
<td>Specifies the state of the message.</td>
</tr>
</tbody>
</table>

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

AvailableMessages

This instance property specifies the number of messages that raised this notification.

Declaration

```csharp
// C#
public int AvailableMessages {get;}
```
OracleAQMessageAvailableEventArgs Class

Property Value
An integer indicating the number of messages that raised this notification.

Remarks
The property value is 1 for a regular notification type. The notification type can be specified using the OracleAQQueue.Notification property.

This property is not relevant if the NotificationType is OracleAQNotificationType.Timeout.

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

ConsumerName
This property provides the name of the consumer for which the message is available for dequeuing.

Declaration
// C#
public string ConsumerName {get;}

Property Value
A string that identifies the name of the consumer.

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

Correlation
This instance property specifies the identification for the message.

Declaration
// C#
public string Correlation {get;}

Property Value
A string that specifies the identification for the message.

Remarks
This property specifies the correlation of the message for which the notification is raised. The consumer can then use this identification to dequeue specific messages by setting the "Correlation" property of the OracleAQDequeueOptions object.
OracleAQMessageAvailableEventArgs Properties

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessageAvailableEventArgs Class
- OracleAQMessageAvailableEventArgs Members
- "Correlation" on page 13

Delay

This instance property specifies the duration, in seconds, after which an enqueued message is available for dequeuing.

Declaration

// C#
public int Delay {get;}

Property Value
An integer that indicates the duration, in seconds, after which an enqueued message is available for dequeuing.

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

DeliveryMode

This instance property specifies the delivery mode of the message.

Declaration

// C#
public OracleAQMessageDeliveryMode DeliveryMode {get;}

Property Value
An OracleAQMessageDeliveryMode enumerated value.

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

EnqueueTime

This instance property specifies the time when the message was enqueued.

Declaration

// C#
public DateTime EnqueueTime {get;}

Property Value
A DateTime object.
OracleAQMessageAvailableEventArgs Class

ExceptionQueue
This instance property specifies the name of the queue that the message is moved to if it cannot be processed successfully.

Declaration
// C#
public string ExceptionQueue {get;}

Property Value
The name of the queue that a message to is moved if it cannot be processed successfully.

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleAQMessageAvailableEventArgs Class
- OracleAQMessageAvailableEventArgs Members

Expiration
This instance property specifies the duration, in seconds, for which an enqueued message is available for dequeuing before expiring.

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

Property Value
An integer that specifies the duration, in seconds, for which an enqueued message is available for dequeuing.

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleAQMessageAvailableEventArgs Class
- OracleAQMessageAvailableEventArgs Members

MessageId
This instance property returns an array of message identifiers.

Declaration
// ADO.NET 2.0: C#
public byte[][] MessageId{get;}

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleAQMessageAvailableEventArgs Class
- OracleAQMessageAvailableEventArgs Members
Property Value
A byte[][] that specifies the message identifiers received as part of the notification.

Remarks
This property specifies the message identifiers of the messages that raise the notification.

The size of the MessageId array is 1 for regular notifications. The size of the MessageId array is 1 for grouping notifications if the notification grouping type is OracleAQNotificationGroupingType.Last.

This property is not relevant if the NotificationType is OracleAQNotificationType.Timeout.

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

NotificationType
This property indicates the type of notification such as regular, grouping, or timeout.

Declaration
// C#
public OracleAQNotificationType NotificationType {get;}

Property Value
An OracleAQNotificationType enum value.

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

OriginalMessageId
This property specifies the ID of the message, in the last queue, that generated this message.

Declaration
// C#
public byte[] OriginalMessageId {get;}

Property Value
A byte[] that specifies the original message ID.
OracleAQMessageAvailableEventArgs Class

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

Priority

This instance property specifies the priority of the message.

Declaration

```csharp
// C#
public int Priority {get;}
```

Property Value

An integer that specifies the priority of the message.

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

QueueName

This property indicates the name of the queue that contains the message to be dequeued.

Declaration

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

Property Value

A string.

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

SenderId

This property identifies the original sender of the message.

Declaration

```csharp
// C#
public OracleAQAgent SenderId {get;}
```

Property Value

An OracleAQAgent object.
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessageAvailableEventArgs Class
- OracleAQMessageAvailableEventArgs Members

State

This instance property specifies the state of the message.

Declaration

// C#
public OracleAQMessageState State {get;}

Property Value

An OracleAQMessageState enumerated value.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQMessageAvailableEventArgs Class
- OracleAQMessageAvailableEventArgs Members
OracleAQMessageAvailableEventHandler Delegate

The OracleAQMessageAvailableEventHandler delegate represents the signature of the method that handles the OracleAQQueue.MessageAvailable event.

Declaration

C# declaration:
```csharp
public delegate void OracleAQMessageAvailableEventHandler (object sender, OracleAQMessageAvailableEventArgs eventArgs);
```

Parameters

- **sender**
  
  The source of the event.

- **eventArgs**

  The OracleAQMessageAvailableEventArgs object that contains the event data.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "MessageAvailable Event" on page 12-79
OracleAQQueue Class

An OracleAQQueue object represents a queue.

Class Inheritance
System.Object
  OracleAQQueue

Declaration
// C#
public class OracleAQQueue : IDisposable

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

Remarks
A queue is a repository of messages and may either be a user queue, or an exception queue. A user queue is for normal message processing. A message is moved from a user queue to an exception queue if it cannot be retrieved and processed for some reason.

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Members
- OracleAQQueue Constructors
- OracleAQQueue Static Methods
- OracleAQQueue Properties
- OracleAQQueue Public Methods
- OracleAQQueue Events
OracleAQQueue Members

OracleAQQueue members are listed in the following tables.

OracleAQQueue Constructors
OracleAQQueue constructors are listed in Table 12–19.

Table 12–19 OracleAQQueue Constructors

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

OracleAQQueue Static Methods
The OracleAQQueue static method is listed in Table 12–20.

Table 12–20 OracleAQQueue Static Methods

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listen</td>
<td>Listens for messages on one or more queues for one or more consumers, as specified in the array of OracleAQAgent objects (Overloaded).</td>
</tr>
</tbody>
</table>

OracleAQQueue Properties
OracleAQQueue properties are listed in Table 12–21.

Table 12–21 OracleAQQueue Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Specifies the OracleConnection object associated with the queue.</td>
</tr>
<tr>
<td>DequeueOptions</td>
<td>Specifies the dequeueing options to use when dequeuing a message from a queue.</td>
</tr>
<tr>
<td>EnqueueOptions</td>
<td>Specifies the enqueueing options used to enqueue a message to a queue.</td>
</tr>
<tr>
<td>MessageType</td>
<td>Specifies the type of queue table associated with this queue.</td>
</tr>
<tr>
<td>Name</td>
<td>Returns the name of the queue.</td>
</tr>
<tr>
<td>Notification</td>
<td>Specifies the various notification options for notifications that are registered using the MessageAvailable event.</td>
</tr>
<tr>
<td>NotificationConsumers</td>
<td>Specifies the array of consumers, for a multiconsumer queue, that are to be notified asynchronously for any incoming messages on the queue.</td>
</tr>
<tr>
<td>UdtTypeName</td>
<td>Specifies the type name on which the queue and the corresponding queue table is based if the MessageType is OracleAQMessageType.UDT.</td>
</tr>
</tbody>
</table>
OracleAQQueue Public Methods

The OracleAQQueue public methods are listed in Table 12–22.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dequeue</td>
<td>Dequeues messages from queues (Overloaded).</td>
</tr>
<tr>
<td>DequeueArray</td>
<td>Dequeues multiple messages from queues (Overloaded).</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases any resources or memory allocated by the object</td>
</tr>
<tr>
<td>Enqueue</td>
<td>Enqueues messages to queues (Overloaded).</td>
</tr>
<tr>
<td>EnqueueArray</td>
<td>Enqueues multiple messages to a queue (Overloaded).</td>
</tr>
<tr>
<td>Listen</td>
<td>Listens for messages on the queue on behalf of listenConsumers (Overloaded).</td>
</tr>
</tbody>
</table>

OracleAQQueue Events

The OracleAQQueue event is listed in Table 12–23.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageAvailable Event</td>
<td>Notifies when a message is available in the queue for NotificationConsumers.</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
OracleAQQueue Constructors

OracleAQQueue constructors create new instances of the OracleAQQueue class.

Overload List:

- **OracleAQQueue(string)**
  This constructor takes a queue name to initialize a queue object.

- **OracleAQQueue(string, OracleConnection)**
  This constructor takes a queue name and connection to initialize a queue object.
  The connection does not need be open during the queue object construction.

- **OracleAQQueue(string, OracleConnection, OracleAQMessageType)**
  This constructor takes a queue name, connection, and message type enumeration to initialize a queue object.

- **OracleAQQueue(string, OracleConnection, OracleAQMessageType, string)**
  This constructor takes a queue name, connection, message type enumeration, and UDT type name to initialize a queue object.

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

OracleAQQueue(string)

This constructor takes a queue name to initialize a queue object.

Declaration

```csharp
// C#
public OracleAQQueue(string name);
```

Parameters

- **name**
  The name of the queue as specified in the database.

Exceptions

- ArgumentNullException - The queue name is null.
- ArgumentException - The queue name is empty.

Remarks

The operation of creating an OracleAQQueue object does not involve checking for the existence of the queue in the database.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
OracleAQQueue(string, OracleConnection)

This constructor takes a queue name and connection to initialize a queue object. The connection does not need to be open during the queue object construction.

Declaration

// C#
public OracleAQQueue(string name, OracleConnection con);

Parameters

- **name**
  
  Name of the queue as specified in the database.

- **con**
  
  An OracleConnection object that connects to the queue.

Exceptions

- ArgumentNullException - Either the connection is null or queue name is null.
- ArgumentException - Queue name is empty.

Remarks

The connection can be accessed using the `Connection` property, and it must be opened before calling any operational APIs such as `Enqueue` and `Dequeue`.

Creating an OracleAQQueue object does not check for the existence of the queue in the database.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members

OracleAQQueue(string, OracleConnection, OracleAQMessageType)

This constructor takes a queue name, connection and message type enumeration to initialize a queue object. The connection does not need to be open during the queue object construction.

Declaration

// C#
public OracleAQQueue(string name, OracleConnection con, OracleAQMessageType messageType);

Parameters

- **name**
  
  The name of the queue as specified in the database.

- **con**
  
  An OracleConnection object that is used to connect to the queue.

- **messageType**
An `OracleAQMessageType` enumeration specifying the type of the message that is enqueued or dequeued from this queue.

**Exceptions**

- `ArgumentNullException` - Either the connection is null or queue name is null.
- `ArgumentException` - Queue name is empty or the specified message type is not valid.

**Remarks**

Creating an `OracleAQQueue` object does not check for the existence of the queue in the database.

You need to set the `UdtTypeName` property before using the queue object if the `messageType` is a UDT. Another approach is to create a queue using the other constructor overload by supplying the `udtTypeName`.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleAQQueue` Class
- `OracleAQQueue Members`

**OracleAQQueue(string, OracleConnection, OracleAQMessageType, string)**

This constructor takes a queue name, connection, message type enumeration, and UDT type name to initialize a queue object. The connection does not need to be open during the queue object construction.

**Declaration**

```csharp
// C#
public OracleAQQueue(string name, OracleConnection con, OracleAQMessageType messageType, string udtTypeName);
```

**Parameters**

- `name`
  The name of the queue as specified in the database.

- `con`
  An `OracleConnection` object that is used to connect to the queue.

- `messageType`
  An `OracleAQMessageType` enumeration specifying the type of the message that is enqueued or dequeued from this queue.

- `udtTypeName`
  The name of the database object type used if the `messageType` is UDT. The `udtTypeName` parameter represents the type on which the queue is based.

**Exceptions**

- `ArgumentNullException` - The connection is null or the queue name is null.
- `ArgumentException` - The queue name is empty or the specified `messageType` is not valid.
Remarks
Creating an OracleAQQueue object does not check for the existence of the queue in the database.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
OracleAQQueue Static Methods

OracleAQQueue static methods are listed in Table 12–24.

Table 12–24  OracleAQQueue Static Methods

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listen</td>
<td>Listens for messages on one or more queues for one or more consumers, as specified in the array of OracleAQAgent objects (Overloaded).</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members

Listen

Listen methods listen for messages on one or more queues for one or more consumers as specified in the array of OracleAQAgent objects.

Overload list

- Listen(OracleConnection, OracleAQAgent[])
  
  This static method listens for messages on one or more queues for one or more consumers as specified in the array of OracleAQAgent objects.

- Listen(OracleConnection, OracleAQAgent[], int)
  
  This static method listens for messages on one or more queues for one or more consumers as specified in the array of OracleAQAgent objects. It also specifies a wait time.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members

Listen(OracleConnection, OracleAQAgent[])

This static method listens for messages on one or more queues for one or more consumers as specified in the array of OracleAQAgent objects.

Declaration

```
// C#
public static OracleAQAgent Listen(OracleConnection con, OracleAQAgent[] listenConsumers);
```

Parameters

- `con`
  
  An OracleConnection instance.

- `listenConsumers`
The array of consumers being listened for. The name of the OracleAQAgent object must be null or empty for single consumer queues.

Return Value
An OracleAQAgent object.

Exceptions
ArgumentNullException - The con or listenConsumers parameter is null.
InvalidOperationException - The connection is not open.

Remarks
Listen is useful in situations where one needs to monitor multiple queues until a message is available for a consumer in one of the queues. The Name property of the OracleAQAgent object represents the name of the consumer and the Address property represents the name of the queue.

This call blocks the calling thread until there is a message ready for consumption for a consumer in the list. It returns an OracleAQAgent object which specifies the consumer and queue for which a message is ready to be dequeued.

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

Listen(OracleConnection, OracleAQAgent[], int)

This static method listens for messages on one or more queues for one or more consumers as specified in the array of OracleAQAgent objects. The Name property of the OracleAQAgent object represents the name of the consumer and the Address property of the OracleAQAgent object represents the name of the queue.

In case of timeout, this method returns null.

Declaration
// C#
public static OracleAQAgent Listen(OracleConnection con, OracleAQAgent[] listenConsumers, int waitTime);

Parameters
- con
  An OracleConnection instance.
- listenConsumers
  The array of consumers being listened for. The name of the OracleAQAgent object must be null or empty for single consumer queues.
- waitTime
  Wait time in seconds.

Return Value
An OracleAQAgent object.
Exceptions
ArgumentNullException - The con or listenConsumers parameter is null.
InvalidOperationException - The connection is not open.
ArgumentException - waitTime is less than -1.

Remarks
Listen is useful in situations where one needs to monitor multiple queues until a message is available for a consumer in one of the queues. The Name property of the OracleAQAgent object represents the name of the consumer and the Address property of the OracleAQAgent object represents the name of the queue.

A waitTime of -1 implies an infinite wait time.

This call blocks the calling thread until there is a message ready for consumption for a consumer in the list. It returns an OracleAQAgent object which specifies the consumer and queue for which a message is ready to be dequeued.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
OracleAQQueue Properties

OracleAQQueue properties are listed in Table 12–25.

### Table 12–25  OracleAQQueue Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Specifies the OracleConnection object associated with the queue.</td>
</tr>
<tr>
<td>DequeueOptions</td>
<td>Specifies the dequeueing options to use when dequeuing a message from a queue.</td>
</tr>
<tr>
<td>EnqueueOptions</td>
<td>Specifies the enqueueing options used to enqueue a message to a queue.</td>
</tr>
<tr>
<td>MessageType</td>
<td>Specifies the type of queue table associated with this queue.</td>
</tr>
<tr>
<td>Name</td>
<td>Returns the name of the queue.</td>
</tr>
<tr>
<td>Notification</td>
<td>Specifies the various notification options for notifications that are registered using the MessageAvailable event.</td>
</tr>
<tr>
<td>NotificationConsumers</td>
<td>Specifies the array of consumers, for a multiconsumer queue, that are to be notified asynchronously for any incoming messages on the queue.</td>
</tr>
<tr>
<td>UdtTypeName</td>
<td>Specifies the type name on which the queue and the corresponding queue table is based if the MessageType is OracleAQMessageType.UDT.</td>
</tr>
</tbody>
</table>

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

**Connection**

This property specifies the OracleConnection object associated with the queue.

**Declaration**

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

**Property Value**

An OracleConnection object that indicates the connection associated with the queue.

**Exceptions**

ObjectDisposedException - The object is already disposed.

**Remarks**

This connection must be opened before calling methods like Enqueue and Dequeue.
DequeueOptions

This instance property specifies the dequeueing options to use when dequeuing a message from a queue.

Declaration

// C#
public OracleAQDequeueOptions DequeueOptions {get; set}

Property Value

An OracleAQDequeueOptions object.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is an OracleAQDequeueOptions object with default property values. Setting this property to null resets all dequeue options to their default values.

See Also:

■ "Oracle.DataAccess.Client Namespace” on page 1-4
■ OracleAQQueue Class
■ OracleAQQueue Members

EnqueueOptions

This instance property specifies the enqueueing options used to enqueue a message to a queue.

Declaration

// C#
public OracleAQEnqueueOptions EnqueueOptions {get; set}

Property Value

An OracleAQEnqueueOptions object.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

The default value is an OracleAQEnqueueOptions object with default property values. Setting this property to null resets all enqueue options to their default values.

See Also:

■ "Oracle.DataAccess.Client Namespace” on page 1-4
■ OracleAQQueue Class
■ OracleAQQueue Members
OracleAQQueue Properties

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

**MessageType**

This instance property specifies the type of queue table associated with this queue.

**Declaration**

```csharp
// C#
public OracleAQMessageType MessageType {get; set;}
```

**Property Value**

An `OracleAQMessageType` enumerated value.

**Exceptions**

- `ArgumentOutOfRangeException` - The type value specified is invalid.
- `ObjectDisposedException` - The object is already disposed.

**Remarks**

The `MessageType` property also dictates the type of message payloads that are enqueued or dequeued from the queue. It is possible to enqueue a variety of payloads depending on the `MessageType`.

Table 12–26 lists the allowed payload types for various message types.

<table>
<thead>
<tr>
<th>OracleAQQueue.MessageType</th>
<th>Allowed OracleAQMessage.Payload type to Enqueue</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleAQMessageType.Raw</td>
<td>OracleBinary, byte[]</td>
</tr>
<tr>
<td>OracleAQMessageType.Xml</td>
<td>OracleXmlType, XmlReader, String (well-formed XML, else exception is raised)</td>
</tr>
<tr>
<td>OracleAQMessageType.UDT</td>
<td>UDT Custom Object</td>
</tr>
</tbody>
</table>

Table 12–27 lists the payload types for dequeued messages.

<table>
<thead>
<tr>
<th>OracleAQQueue.MessageType</th>
<th>DequeueOptions.ProviderSpecificType</th>
<th>OracleAQMessage.Payload of dequeued message</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleAQMessageType.Xml</td>
<td>true</td>
<td>OracleXmlType</td>
</tr>
<tr>
<td>OracleAQMessageType.Xml</td>
<td>false</td>
<td>XmlReader</td>
</tr>
<tr>
<td>OracleAQMessageType.Raw</td>
<td>true</td>
<td>OracleBinary</td>
</tr>
<tr>
<td>OracleAQMessageType.Raw</td>
<td>false</td>
<td>Byte[]</td>
</tr>
<tr>
<td>OracleAQMessageType.UDT</td>
<td>N.A.</td>
<td>UDT Custom Object</td>
</tr>
</tbody>
</table>
OracleAQQueue Class

Name

This instance property returns the name of the queue.

Declaration

// C#
public string Name {get;}

Property Value

A string that indicates the name of the queue.

Exceptions

ObjectDisposedException - The object is already disposed.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members

Notification

This instance property specifies the various notification options for notifications that are registered using the MessageAvailable event.

Declaration

// C#
public OracleNotificationRequest Notification {get;}

Property Value

Specifies an OracleNotificationRequest object whose properties can be changed to alter the notification behavior.

Remarks

This property can be used to change various notification options. The notification options must be changed before registering with the MessageAvailable event. This property can be modified again only after unregistering from the MessageAvailable event.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
NotificationConsumers
This instance property specifies the array of consumers, for a multiconsumer queue, that are to be notified asynchronously for any incoming messages on the queue.

Declaration
// C#
public string[] NotificationConsumers {get; set;}

Property Value
Specifies an array of consumer name strings for which the notifications are delivered.

Exceptions
ObjectDisposedException - The object is already disposed.
InvalidOperationException - MessageAvailable registration is active.

Remarks
The consumer names must be in uppercase. This functionality only supports queues with uppercase names.

The list of consumers is used in the MessageAvailable event. The list must be set before registering for the event. This property cannot be modified after registering for the MessageAvailable event. This property can be modified again only after unregistering from MessageAvailable event.

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

UdtTypeName
This instance property specifies the type name on which the queue and the corresponding queue table is based if the MessageType is OracleAQMessageType.UDT.

Declaration
// C#
public string UdtTypeName {get; set;}

Property Value
Specifies the Oracle user-defined type name if the MessageType is OracleAQMessageType.UDT.

Exceptions
ObjectDisposedException - The object is already disposed.

Remarks
The UdtTypeName property corresponds to the user-defined type name of the payload. This property must always be specified if the payload is a user-defined type. This property need not be set for other payload types.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
OracleAQQueue Public Methods

OracleAQQueue public methods are listed in Table 12–28.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dequeue</td>
<td>Dequeues messages from queues (Overloaded).</td>
</tr>
<tr>
<td>DequeueArray</td>
<td>Dequeues multiple messages from queues (Overloaded).</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases any resources or memory allocated by the object</td>
</tr>
<tr>
<td>Enqueue</td>
<td>Enqueues messages to queues (Overloaded).</td>
</tr>
<tr>
<td>EnqueueArray</td>
<td>Enqueues multiple messages to a queue (Overloaded).</td>
</tr>
<tr>
<td>Listen</td>
<td>Listens for messages on the queue on behalf of listenConsumers (Overloaded).</td>
</tr>
</tbody>
</table>

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

Dequeue

Dequeue methods dequeue messages from queues.

Overload List
- Dequeue()
  This instance method dequeues messages from a queue using the DequeueOptions for the instance.
- Dequeue(OracleAQDequeueOptions)
  This instance method dequeues messages from a queue using the supplied dequeue options.

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

Dequeue()

This instance method is used to dequeue a message from a queue using the DequeueOptions for the instance.

Declaration

// C#
public OracleAQMessage Dequeue();

**Return Value**
An OracleAQMessage instance representing the dequeued message.

**Exceptions**
- InvalidOperationException - The connection is not open.
- ObjectDisposedException - The object is already disposed.
- OracleException - In case of timeout, an exception is thrown with the message, ORA-25228: timeout or end-of-fetch during message dequeue from queue_name. Timeout may happen if DequeueOptions.Wait is set to a value other than -1.

**Remarks**
The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

Dequeued buffered messages always have null MessageId values.

**Dequeue(OracleAQDequeueOptions)**
This instance method dequeues messages from a queue using the supplied dequeue options.

**Declaration**
// C#
public OracleAQMessage Dequeue(OracleAQDequeueOptions dequeueOptions);

**Parameters**
- dequeueOptions
  An OracleAQDequeueOptions object.

**Return Value**
An OracleAQMessage instance representing the dequeued message.

**Exceptions**
- InvalidOperationException - The connection is not open.
- ObjectDisposedException - The object is already disposed.
- OracleException - In case of timeout, an exception is thrown with the message, ORA-25228: timeout or end-of-fetch during message dequeue from queue_name. Timeout may happen if DequeueOptions.Wait is set to a value other than -1.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
Remarks
If the supplied dequeueOptions object is null, then the dequeue options default values are used. The queue object’s DequeueOptions property is ignored for this operation.

Calling this method does not change the DequeueOptions property of the queue.

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

Dequeued buffered messages always have null MessageId values.

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

**DequeueArray**

DequeueArray methods dequeue multiple messages from queues.

**Overload List**
- DequeueArray(int)
  
  This instance method dequeues multiple messages from a queue using the DequeueOptions of the instance.

- DequeueArray(int, OracleAQDequeueOptions)
  
  This instance method dequeues multiple messages from a queue using the supplied dequeue options.

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

**DequeueArray(int)**

This instance method dequeues multiple messages from a queue using the DequeueOptions of the instance.

**Declaration**

```csharp
// C#
public OracleAQMessage[] DequeueArray(int dequeueCount);
```

**Parameters**

- dequeueCount

  An integer specifying the numbers of messages to dequeue.

**Return Value**

An array of OracleAQMessage instances representing the dequeued messages.
Exceptions
ArgumentOutOfRangeException - dequeueCount is less than or equal to 0.
InvalidOperationException - The connection is not open.
ObjectDisposedException - The object is already disposed.
OracleException - In case of timeout, an exception is thrown with the message, ORA-25228: timeout or end-of-fetch during message dequeue from queue_name. Timeout may happen if DequeueOptions.Wait is set to a value other than -1.

Remarks
This method is supported for Oracle Database 10g and higher releases.
The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must be set as well.
The size of the returned array may be less than the dequeueCount. It depends on the actual number of messages present in the queue.
For database versions earlier than Oracle Database 11g Release 2 (11.2.0.3), the MessageId property of persistent OracleAQMessage objects retrieved using DequeueArray is always null.
Dequeued buffered messages always have null MessageId values irrespective of the database version.

See Also:
  ■ "Oracle.DataAccess.Client Namespace" on page 1-4
  ■ OracleAQQueue Class
  ■ OracleAQQueue Members

DequeueArray(int, OracleAQDequeueOptions)
This instance method dequeues multiple messages from a queue using the supplied dequeue options.

Declaration
// C#
public OracleAQMessage[] DequeueArray(int dequeueCount, OracleAQDequeueOptions dequeueOptions);

Parameters
  ■ dequeueCount
    An integer specifying the numbers of messages to dequeue.
  ■ dequeueOptions
    An OracleAQDequeueOptions object.

Return Value
An array of OracleAQMessage instances representing the dequeued messages.

Exceptions
ArgumentOutOfRangeException - dequeueCount is less than or equal to 0.
InvalidOperation exception - The connection is not open.
ObjectDisposed exception - The object is already disposed.
OracleException - In case of timeout, an exception is thrown with the message, ORA-25228: timeout or end-of-fetch during message dequeue from queue_name. Timeout may happen if DequeueOptions.Wait is set to a value other than -1.

Remarks
This method is supported for Oracle Database 10g Release 1 (10.1) and higher releases. Calling this method does not change the DequeueOptions property of the queue.
If the supplied dequeueOptions object is null, then the dequeue options default values are used. The DequeueOptions property of the queue object is ignored in this operation.
The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType. UDT, then the UdtTypeName property must be set as well.
The size of the returned array may be less than the dequeueCount. It depends on the actual number of messages present in the queue.
For database versions earlier than Oracle Database 11g Release 2 (11.2.0.3), the MessageId property of persistent OracleAQMessage objects retrieved using DequeueArray is always null.
Dequeued buffered messages always have null MessageId values irrespective of the database version.

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

Dispose
This method releases any resources or memory allocated by the object.

Declaration
// C#
public void Dispose();

Implements
IDisposable.

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

Enqueue
Enqueue instance methods enqueue messages to queues.
Overload List

- **Enqueue(OracleAQMessage)**
  This instance method enqueues messages to a queue using the EnqueueOptions of the instance.

- **Enqueue(OracleAQMessage, OracleAQEnqueueOptions)**
  This instance method enqueues messages to a queue using the supplied enqueue options.

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

**Enqueue(OracleAQMessage)**

This instance method enqueues messages to a queue using the EnqueueOptions of the instance.

**Declaration**

```csharp
// C#
public void Enqueue(OracleAQMessage message);
```

**Parameters**

- message
  An OracleAQMessage object.

**Exceptions**

- ObjectDisposedException - The object is already disposed.
- InvalidOperationException - The connection is not open.
- ArgumentNullException - The message parameter is null.
- ArgumentException - The message payload is OracleXmlType and the connection used to create OracleXmlType is different from the queue's connection.

**Remarks**

MessageId of the enqueued message is populated after the call to Enqueue completes. Enqueued buffered messages always have null MessageId values.

The MessageType property needs to be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must be set as well.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
Enqueue(OracleAQMessage, OracleAQEnqueueOptions)

This instance method enqueues messages to a queue using the supplied enqueue options.

Declaration

// C#
public void Enqueue(OracleAQMessage message, OracleAQEnqueueOptions enqueueOptions);

Parameters

- **message**
  An OracleAQMessage object.
- **enqueueOptions**
  An OracleAQEnqueueOptions object.

Exceptions

- **ObjectDisposedException** - The object is already disposed.
- **InvalidOperationException** - The connection is not open.
- **ArgumentNullException** - The message parameter is null.
- **ArgumentException** - The message payload is OracleXmlType and the connection used to create OracleXmlType is different from the queue's connection.

Remarks

If the supplied enqueueOptions object is null, then the enqueue options default values are used. The EnqueueOptions property of the queue object is ignored in this operation.

The MessageId of the enqueued message is populated after the call to Enqueue completes. Enqueued buffered messages always have null MessageId values. Calling this method does not change the EnqueueOptions property of the queue.

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members

EnqueueArray

EnqueueArray instance methods enqueue multiple messages to a queue.

Overload List

- **EnqueueArray(OracleAQMessage[])**
  This instance method enqueues multiple messages to a queue using the EnqueueOptions of the instance.

- **EnqueueArray(OracleAQMessage[ ], OracleAQEnqueueOptions)**
This instance method enqueues multiple messages to a queue using the supplied enqueue options.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members

### EnqueueArray(OracleAQMessage[ ]) 
This instance method enqueues multiple messages to a queue using the EnqueueOptions of the instance.

**Declaration**

// C#
public int EnqueueArray(OracleAQMessage[] messages);

**Parameters**
- *messages*

  An array of OracleAQMessage objects.

**Return Value**
An integer representing the number of messages actually enqueued.

**Exceptions**
- ArgumentNullException - The message parameter is null.
- ArgumentException - At least one of the OracleAQMessage[] elements is null, or at least one of the OracleAQMessage[] elements has a payload of OracleXmlType, which is created using a connection that is different from the queue's connection.
- InvalidOperationException - The OracleAQMessage array is empty or the connection is not open.
- ObjectDisposedException - The object is already disposed.

**Remarks**
This method is supported by Oracle Database 10g and higher releases. The MessageId properties of the enqueued messages are populated after the call to Enqueue completes. Enqueued buffered messages always have null MessageId values.

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
EnqueueArray(OracleAQMessage[], OracleAQEnqueueOptions)

This instance method enqueues multiple messages to a queue using the supplied enqueue options.

Declaration

// C#
public int EnqueueArray(OracleAQMessage[] messages, OracleAQEnqueueOptions enqueueOptions);

Parameters

- messages
  An array of OracleAQMessage objects.
- enqueueOptions
  An OracleAQEnqueueOptions object.

Return Value

An integer representing the number of messages actually enqueued.

Exceptions

ArgumentNullException - The message parameter is null.
ArgumentException - At least one of the OracleAQMessage[] elements is null, or at least one of the OracleAQMessage[] elements has a payload of OracleXmlType, which is created using a connection that is different from the queue’s connection.
InvalidOperationException - The OracleAQMessage array is empty or the connection is not open.
ObjectDisposedException - The object is already disposed.

Remarks

This method is supported by Oracle Database 10g and higher releases. MessageId properties of the enqueued messages are populated after the call to Enqueue completes. Enqueued buffered messages always have null MessageId values. Calling this method does not change the EnqueueOptions property of the queue.

If the supplied enqueueOptions object is null, then the enqueue options default values are used. The EnqueueOptions property of the queue object is ignored in this operation.

The MessageType property must be set appropriately before calling this function. If the MessageType is OracleAQMessageType.UDT, then the UdtTypeName property must also be set.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members

Listen

Listen methods listen for messages on the queue on behalf of listenConsumers.
Overload List

- **Listen(string[])**
  This method listens for messages on the queue on behalf of listenConsumers.

- **Listen(string[], int)**
  This method listens for messages on behalf of listenConsumers for a specified time.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members

### Listen(string[]) Method

This method listens for messages on the queue on behalf of listenConsumers.

#### Declaration

```csharp
public string Listen(string[] listenConsumers);
```

#### Parameters

- **listenConsumers**
  An array of consumers to listen for on this queue. This parameter should be null in case of single consumer queues.

#### Return Value

A string.

#### Exceptions

- `InvalidOperationException` - The connection is not open.
- `ObjectDisposedException` - The object is already disposed.

#### Remarks

This call blocks the calling thread until there is a message ready for consumption for a consumer in the listenConsumers array. It returns a string representing the consumer name for which the message is ready.

Listen is useful in situations that require waiting until a message is available in the queue for consumers whose names are specified in listenConsumers.

#### Example

The following example demonstrates using the `Listen` method. The first part of the example performs the requisite database setup for the database user, SCOTT. The second part of the example demonstrates how a thread can listen and wait until a message is enqueued.

```sql
-- Part I: Database setup required for this demo

-- SQL to grant appropriate privilege to database user, SCOTT
```

---

---
SQL> ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY Pwd4Sct;
User altered.
GRANT ALL ON DBMS_AQADM TO scott;

-- PL/SQL to create queue-table and queue and start queue for SCOTT
BEGIN
  DBMS_AQADM.CREATE_QUEUE_TABLE(
    queue_table=>'scott.test_q_tab',
    queue_payload_type=>'RAW',
    multiple_consumers=>FALSE);
  DBMS_AQADM.CREATE_QUEUE(
    queue_name=>'scott.test_q',
    queue_table=>'scott.test_q_tab');
  DBMS_AQADM.START_QUEUE(queue_name=>'scott.test_q');
END;
/

-- PL/SQL to stop queue and drop queue & queue-table from SCOTT
BEGIN
  DBMS_AQADM.STOP_QUEUE('scott.test_q');
  DBMS_AQADM.DROP_QUEUE(
    queue_name => 'scott.test_q',
    auto_commit => TRUE);
  DBMS_AQADM.DROP_QUEUE_TABLE(
    queue_table => 'scott.test_q_tab',
    force => FALSE,
    auto_commit => TRUE);
END;
/

-- End of Part I, database setup.

// Part II: Demonstrates using the Listen method
// C#
using System;
using System.Text;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;
using System.Threading;

namespace ODPSample
{
  /// <summary>
  /// Demonstrates how a thread can listen and wait until a message is enqueued.
  /// Once a message is enqueued, the listening thread returns from the
  /// blocked Listen() method invocation and dequeues the message.
  /// </summary>
  class EnqueueDequeue
  {
    static bool s_bListenReturned = false;
    static void Main(string[] args)
    {

// Create connection
string constr = "user id=scott;password=Pwd4Sct;data source=oracle";
OracleConnection con = new OracleConnection(constr);

// Create queue
OracleAQQueue queue = new OracleAQQueue("scott.test_q", con);

try
{
    // Open connection
    con.Open();

    // Set message type for the queue
    queue.MessageType = OracleAQMessageType.Raw;

    // Spawning a thread which will listen for a message
    ThreadStart ts = new ThreadStart(TestListen);
    Thread t = new Thread(ts);
    t.Start();
    System.Threading.Thread.Sleep(2000);

    // Begin transaction for enqueue
    OracleTransaction txn = con.BeginTransaction();

    // Prepare message and RAW payload
    OracleAQMessage enqMsg = new OracleAQMessage();
    byte[] bytePayload = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    enqMsg.Payload = bytePayload;

    // Prepare to Enqueue
    queue.EnqueueOptions.Visibility = OracleAQVisibilityMode.OnCommit;
    Console.WriteLine("[Main Thread] Enqueuing a message...");
    Console.WriteLine("[Main Thread] Enqueued Message Payload : " + ByteArrayToString(enqMsg.Payload as byte[]));
    Console.WriteLine();

    // Enqueue message
    queue.Enqueue(enqMsg);

    // Enqueue transaction commit
    txn.Commit();

    // Loop till Listen returns
    while (!s_bListenReturned)
    {
        System.Threading.Thread.Sleep(1000);
    }
}
catch (Exception e)
{
    Console.WriteLine("Error: {0}", e.Message);
}
finally
{
    // Close/Dispose objects
    queue.Dispose();
    con.Close();
    con.Dispose();
}

}
static void TestListen()
{
    // Create connection
    string constr = "user id=scott;password=Pwd4Sct;data source=oracle"
    OracleConnection conListen = new OracleConnection(constr);

    // Create queue
    OracleAQQueue queueListen = new OracleAQQueue("scott.test_q", conListen);

    try
    {
        // Open the connection for Listen thread.
        // Connection blocked on Listen thread can not be used for other DB
        // operations
        conListen.Open();

        // Set message type for the queue
        queueListen.MessageType = OracleAQMessageType.Raw;

        // Listen
        queueListen.Listen(null);

        Console.WriteLine("[Listen Thread] Listen returned... Dequeuing...");

        // Begin txn for Dequeue
        OracleTransaction txn = conListen.BeginTransaction();

        // Prepare to Dequeue
        queueListen.DequeueOptions.Wait = 10;

        // Dequeue message
        OracleAQMessage deqMsg = queueListen.Dequeue();
        Console.WriteLine("[Listen Thread] Dequeued Message Payload : " + ByteArrayToString(deqMsg.Payload as byte[]));

        // Dequeue txn commit
        txn.Commit();

        // Allow the main thread to exit
        s_bListenReturned = true;
    }
    catch (Exception e)
    {
        Console.WriteLine("Error: {0}", e.Message);
    }
    finally
    {
        // Close/Dispose objects
        queueListen.Dispose();
        conListen.Close();
        conListen.Dispose();
    }
}

// Function to convert byte[] to string
static private string ByteArrayToString(byte[] byteArray)
{
    StringBuilder sb = new StringBuilder();

for (int n = 0; n < byteArray.Length; n++)
{
    sb.Append((int.Parse(byteArray[n].ToString())).ToString("X"));
}
return sb.ToString();
}

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

Listen (string[], int)
This method listens for messages on behalf of listenConsumers for a specified time.

Declaration
// C#
public string Listen(string[] listenConsumers, int waitTime);

Parameters
- listenConsumers
  Array of consumers for which to listen on this queue.
- waitTime
  Wait time in seconds.

Return Value
A string

Exceptions
InvalidOperationException - The connection is not open.
ArgumentException - waitTime is less than -1.
ObjectDisposedException - The object is already disposed.

Remarks
Listen is useful in situations that require waiting until a message is available in the queue for consumers whose names are specified in listenConsumers.

This call blocks the calling thread until there is a message ready for consumption for a consumer in the listenConsumers array. It returns a string representing the consumer name for which the message is ready. The method returns null if a timeout occurs.

The listenConsumers parameter should be null for single consumer queues. An empty string is returned in such cases.

A waitTime of -1 implies infinite wait time.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
OracleAQQueue Events

The OracleAQQueue event is listed in Table 12–29.

Table 12–29 OracleAQQueue Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageAvailable</td>
<td>Notifies when a message is available in the queue for NotificationConsumers.</td>
</tr>
</tbody>
</table>

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

MessageAvailable Event

This event is notified when a message is available in the queue for NotificationConsumers.

Declaration

// C#
public event OracleAQMessageAvailableEventHandler MessageAvailable;

Event Data

The event handler receives an OracleAQMessageAvailableEventArgs object.

Exceptions

InvalidOperationException - The connection is not open.

Remarks

Asynchronous notification is supported in all queue tables created with a database compatibility level of 8.1 or higher.

In order to receive the notification about message availability, the client must create an OracleAQMessageAvailableEventHandler delegate to listen to this event. The delegate should be added to this event only after setting the NotificationConsumers and Notification properties.

The notification registration takes place after the first delegate is added to the event. The notification is unregistered when the last delegate is removed from the event. Notifications set on an OracleAQQueue object get cancelled automatically when the object gets disposed.

Oracle Data Provider for .NET opens a port to listen for notifications. HA events, load balancing, and database change notification features also share the same port. This port can be configured centrally by setting the database notification port in an application or Web configuration file. The following example code specifies a port number of 1200:

```xml
<configuration>
  <oracle.dataaccess.client>
    <settings>
      <configuration>
If the configuration file does not exist or the db notification port is not specified, then ODP.NET uses a valid and random port number. The configuration file may also request for a random port number by specifying a db notification port value of -1.

The notification listener, which runs in the same application domain as ODP.NET, uses the specified port number to listen to notifications from the database. A notification listener gets created when the application registers with OracleAQQueue.MessageAvailable event. One notification listener can listen to all notification types. Only one notification listener is created for each application domain.

**Example**

The following example demonstrates application notification. The first part of the example performs the requisite database setup for the database user, SCOTT. The second part of the example demonstrates how an application is notified when a message is available in the queue.

```
-- Part I: Database setup required for this demo

-- SQL to grant appropriate privilege to database user, SCOTT
SQL> ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY Pwd4Sct;
User altered.
SQL> GRANT ALL ON DBMS_AQADM TO scott;

-- PLSQL to create queue-table and queue and start queue for SCOTT
BEGIN
  DBMS_AQADM.CREATE_QUEUE_TABLE(
    queue_table=>'scott.test_q_tab',
    queue_payload_type=>'RAW',
    multiple_consumers=>FALSE);
  DBMS_AQADM.CREATE_QUEUE(
    queue_name=>'scott.test_q',
    queue_table=>'scott.test_q_tab');
  DBMS_AQADM.START_QUEUE(queue_name=>'scott.test_q');
END;
/

-- PLSQL to stop queue and drop queue & queue-table from SCOTT
BEGIN
  DBMS_AQADM.STOP_QUEUE('scott.test_q');
  DBMS_AQADM.DROP_QUEUE(
    queue_name => 'scott.test_q',
    auto_commit => TRUE);
  DBMS_AQADM.DROP_QUEUE_TABLE(
```
queue_table => 'scott.test_q_tab',
force => FALSE,
auto_commit => TRUE);
END;
/
-- End of Part I, database setup.

// Part II: Demonstrates application notification
// C#
using System;
using System.Text;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;
namespace ODPSample
{
    /// <summary>
    /// Demonstrates how the application can be notified when a message is
    /// available in a queue.
    /// </summary>
    class Notification
    {
        static bool isNotified = false;

        static void Main(string[] args)
        {
            // Create connection
            string constr = "user id=scott;password=Pwd4Sct;data source=oracle";
            OracleConnection con = new OracleConnection(constr);

            // Create queue
            OracleAQQueue queue = new OracleAQQueue("scott.test_q", con);

            try
            {
                // Open connection
                con.Open();

                // Set message type for the queue
                queue.MessageType = OracleAQMessageType.Raw;

                // Add the event handler to handle the notification. The
                // MsgReceived method will be invoked when a message is enqueued
                queue.MessageAvailable +=
                    new OracleAQMessageAvailableEventHandler(Notification.MsgReceived);

                Console.WriteLine("Notification registered...");

                // Begin txn for enqueue
                OracleTransaction txn = con.BeginTransaction();

                Console.WriteLine("Now enqueuing message...");

                // Prepare message and RAW payload
                OracleAQMessage enqMsg = new OracleAQMessage();
                byte[] bytePayload = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
                enqMsg.Payload = bytePayload;

                // Prepare to Enqueue
                queue.EnqueueOptions.Visibility = OracleAQVisibilityMode.OnCommit;

                // Enqueue the message
                queue.Enqueue(enqMsg);

                // Mark transaction as committed
                txn.Commit();

                // Notify the application
                if (isNotified) { return; }

                // Notify the application
                queue.MessageAvailable -=
                    new OracleAQMessageAvailableEventHandler(Notification.MsgReceived);

                Console.WriteLine("Message notified.");
            }
            finally
            {
                // Clean up
                queue.Dispose();
            }
        }
    }
}
// Enqueue message
queue.Enqueue(enqMsg);

Console.WriteLine("Enqueued Message Payload : "+ByteArrayToString(enqMsg.Payload as byte[]));
Console.WriteLine("MessageId of Enqueued Message : "+ByteArrayToString(enqMsg.MessageId));
Console.WriteLine();

// Enqueue txn commit
txn.Commit();

// Loop while waiting for notification
while (isNotified == false)
{
    System.Threading.Thread.Sleep(2000);
}
catch (Exception e)
{
    Console.WriteLine("Error: (0)", e.Message);
}
finally
{
    // Close/Dispose objects
    queue.Dispose();
    con.Close();
    con.Dispose();
}
}

static void MsgReceived(object src, OracleAQMessageAvailableEventArgs arg)
{
    try
    {
        Console.WriteLine("Notification Received...");
        Console.WriteLine("QueueName : {0}", arg.QueueName);
        Console.WriteLine("Notification Type : {0}", arg.NotificationType);

        //following type-cast to 'byte[]' is required only for .NET 1.x
        byte[] notifiedMsgId = (byte[])arg.MessageId[0];
        Console.WriteLine("MessageId of Notified Message : "+ByteArrayToString(notifiedMsgId));
        isNotified = true;
    }
    catch (Exception e)
    {
        Console.WriteLine("Error: (0)", e.Message);
    }
}

// Function to convert byte[] to string
static private string ByteArrayToString(byte[] byteArray)
{
    StringBuilder sb = new StringBuilder();
    for (int n = 0; n < byteArray.Length; n++)
    {
        sb.Append((int.Parse(byteArray[n].ToString())).ToString("X"));
    }
    return sb.ToString();
}
return sb.ToString();
}
}

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleAQQueue Class
- OracleAQQueue Members
OracleAQDequeueMode Enumeration

Table 12–30 lists all the OracleAQDequeueMode enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse</td>
<td>Reads the message without acquiring any lock on the message. This is equivalent to a SELECT statement.</td>
</tr>
<tr>
<td>Locked</td>
<td>Reads and obtains a write lock on the message. The lock lasts for the duration of the transaction. This is equivalent to a SELECT FOR UPDATE statement.</td>
</tr>
<tr>
<td>Remove</td>
<td>Reads the message and updates or deletes it. This is the default. The message can be retained in the queue table based on the retention properties</td>
</tr>
<tr>
<td>RemoveNoData</td>
<td>Confirms receipt of the message but does not deliver the actual message content.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleAQDequeueOptions Class" on page 12-8
- "DequeueMode" on page 12-14
OracleAQMessageDeliveryMode Enumeration

The `OracleAQMessageDeliveryMode` enumeration type specifies the delivery mode of the message. 

Table 12–31 lists all the `OracleAQMessageDeliveryMode` enumeration values with a description of each enumerated value.

### Table 12–31 OracleAQMessageDeliveryMode Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Buffered**                | Indicates a buffered message. | Both enqueue and dequeue buffered messaging operations must be in IMMEDIATE visibility mode. This means that these operations cannot be part of another transaction. You cannot specify delay when enqueuing buffered messages.  

Dequeueing applications can choose to dequeue persistent messages only, buffered messages only, or both types.  

Buffered messages can be queried using the `AQ$Queue_Table_Name` view. These messages appear with states, IN–MEMORY or SPILLED.  

Buffered messaging is supported in Oracle Database 10g release 2 (10.2) and higher releases. Recipient lists are supported for buffered messaging.  

Buffered messaging is supported in all queue tables created with a database compatibility level of 8.1 or higher. Transaction grouping queues and array enqueues are not supported for buffered messages in Oracle Database 11g release 1 (11.1). One can still use the array enqueue procedure to enqueue buffered messages, but the array size must be set to 1. Array dequeue is not supported for buffered messaging, but one can still use the array dequeue procedure by setting array size to 1.  

Buffered messaging is faster than persistent messaging. Use buffered messaging for applications that do not require the reliability and transaction support of Oracle Streams AQ persistent messaging.

| **Persistent**              | Indicates a persistent message. | Persistent messaging ensures reliability and support transactions. It is slower than buffered messaging. |

| **PersistentOrBuffered**   | Indicates a persistent or buffered message. | This is used with `Dequeue()` when a consumer wants to dequeue a message irrespective of whether it is Persistent or Buffered. |

**Requirements**

- **Namespace:** `Oracle.DataAccess.Client`
- **Assembly:** `Oracle.DataAccess.dll`
- **ODP.NET Version:** ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleAQDequeueOptions Class" on page 12-8
- "DeliveryMode" on page 12-13
OracleAQMessageState Enumeration

The OracleAQMessageState enumeration type identifies the state of the message at the time of dequeue.

Table 12–32 lists all the OracleAQMessageState enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expired</td>
<td>Indicates that the message has been moved to the exception queue.</td>
</tr>
<tr>
<td>Processed</td>
<td>Indicates that the message has been processed and retained.</td>
</tr>
<tr>
<td>Ready</td>
<td>Indicates that the message is ready to be processed.</td>
</tr>
<tr>
<td>Waiting</td>
<td>Indicates that the message delay has not been reached.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleAQMessage Class" on page 12-25
- "State" on page 12-36
OracleAQMessageType Enumeration

The OracleAQMessageType enumeration type specifies the message payload type. Table 12–33 lists all the OracleAQMessageType enumeration values with a description of each enumerated value.

Table 12–33  OracleAQMessageType Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>Indicates the Raw message type. The data type of the payload must be either OracleBinary or byte[] to enqueue the message.</td>
</tr>
<tr>
<td>UDT</td>
<td>Indicates the Oracle UDT message type. The ODP.NET AQ implementation currently does not support user defined types with LOB attributes. It also does not support other variants of user defined types such as VARRAY and nested tables, as Oracle Streams AQ does not support them inherently.</td>
</tr>
<tr>
<td>XML</td>
<td>Indicates the XML message type. The data type of the payload must be OracleXmlType, XmlReader, or String in order to enqueue the message. If the data type is String, it must be well-formed XML, else an exception is raised when enqueuing the message.</td>
</tr>
</tbody>
</table>

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleAQQueue Class" on page 12-48
- "OracleAQQueue Constructors" on page 12-51
- "MessageType" on page 12-60
OracleAQNavigationMode Enumeration

Table 12–34 lists all the OracleAQNavigationMode enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstMessage</td>
<td>Retrieves the first message that is available and matches the search criteria. This resets the position to the beginning of the queue.</td>
</tr>
<tr>
<td>FirstMessageMultiGroup</td>
<td>Indicates that a call to DequeueArray resets the position to the beginning of the queue, and dequeues messages that are available and match the search criteria. Messages are dequeued till the dequeueCount limit is reached. The dequeued messages can belong to different transaction groups. You can use the OracleAQMessage.TransactionGroup property to distinguish between messages from different transaction groups. All messages from the same transaction group have the same value for the OracleAQMessage.TransactionGroup property.</td>
</tr>
<tr>
<td>NextMessage</td>
<td>Retrieves the next message that is available and matches the search criteria. If the previous message belongs to a message group, AQ retrieves the next available message that matches the search criteria and belongs to the message group.</td>
</tr>
<tr>
<td>NextMessageMultiGroup</td>
<td>Indicates that a call to DequeueArray dequeues the next set of messages that are available and match the search criteria. Messages are dequeued till the dequeueCount limit is reached. The dequeued messages can belong to different transaction groups. You can use the OracleAQMessage.TransactionGroup property to distinguish between messages from different transaction groups. All messages from the same transaction group have the same value for the OracleAQMessage.TransactionGroup property.</td>
</tr>
<tr>
<td>NextTransaction</td>
<td>Skips the remainder of the current transaction group (if any) and retrieves the first message of the next transaction group.</td>
</tr>
</tbody>
</table>

Requirements

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- "OracleAQDequeueOptions Class” on page 12-8
- "NavigationMode” on page 12-15
The `OracleAQNotificationGroupingType` enumeration type specifies the notification grouping type.

Table 12–35 lists all the `OracleAQNotificationGroupingType` enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last</td>
<td>Indicates that only details of the last message in the notification group are provided.</td>
</tr>
<tr>
<td>Summary</td>
<td>Indicates that the Summary of all messages in the notification group is provided.</td>
</tr>
</tbody>
</table>

**Requirements**

Namespace: `Oracle.DataAccess.Client`

Assembly: `Oracle.DataAccess.dll`

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleNotificationRequest Class" on page 9-22
OracleAQNotificationType Enumeration

The OracleAQNotificationType enumeration type specifies the notification type of the received notification.

Table 12–36 lists all the OracleAQNotificationType enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Indicates that the received notification is a grouping notification.</td>
</tr>
<tr>
<td>Regular</td>
<td>Indicates that the received notification is a regular notification.</td>
</tr>
<tr>
<td>Timeout</td>
<td>Indicates that the received notification is raised due to a timeout.</td>
</tr>
</tbody>
</table>

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

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

Table 12–37 lists all the OracleAQVisibilityMode enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td>Indicates that the enqueue or dequeue operation is not part of the current transaction. The operation constitutes a transaction of its own.</td>
</tr>
<tr>
<td>OnCommit</td>
<td>Indicates that the enqueue or dequeue operation is part of the current transaction. This is the default case.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

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

Class Inheritance
System.Object
    System.MarshalByRefObject
    System.IO.Stream
    Oracle.DataAccess.Types.OracleBFile

Declaration
// C#
public sealed class OracleBFile : Stream, ICloneable, INullable

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 later.

Example
// Database Setup, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has 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 previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // Note that the byte representation of the ABCDABC is 65666768656667

        string restr = "User Id=scott;Password=tiger;Data Source=oracle";
        OracleConnection con = new OracleConnection(restr);
        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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
    ■ "Oracle.DataAccess.Types Namespace" on page 1-9
    ■ 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 13–1.

<table>
<thead>
<tr>
<th>Table 13–1 OracleBFile Constructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor</td>
</tr>
<tr>
<td>OracleBFile Constructors</td>
</tr>
</tbody>
</table>

**OracleBFile Static Fields**

OracleBFile static fields are listed in Table 13–2.

<table>
<thead>
<tr>
<th>Table 13–2 OracleBFile Static Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
</tr>
<tr>
<td>MaxSize</td>
</tr>
<tr>
<td>Null</td>
</tr>
</tbody>
</table>

**OracleBFile Static Methods**

OracleBFile static methods are listed in Table 13–3.

<table>
<thead>
<tr>
<th>Table 13–3 OracleBFile Static Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
</tr>
<tr>
<td>Equals</td>
</tr>
</tbody>
</table>

**OracleBFile Instance Properties**

OracleBFile instance properties are listed in Table 13–4.

<table>
<thead>
<tr>
<th>Table 13–4 OracleBFile Instance Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
</tr>
<tr>
<td>CanRead</td>
</tr>
<tr>
<td>CanSeek</td>
</tr>
<tr>
<td>CanWrite</td>
</tr>
<tr>
<td>Connection</td>
</tr>
<tr>
<td>DirectoryName</td>
</tr>
<tr>
<td>FileExists</td>
</tr>
<tr>
<td>FileName</td>
</tr>
<tr>
<td>IsEmpty</td>
</tr>
</tbody>
</table>
Table 13–4  (Cont.) OracleBFile Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the current instance has a null value</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>

OracleBFile Instance Methods

OracleBFile instance methods are listed in Table 13–5.

Table 13–5  OracleBFile Instance Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Inherited from System.IO.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 System.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 System.IO.Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.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 System.IO.Stream</td>
</tr>
<tr>
<td>Methods</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</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 System.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 1-9
- OracleBFile Members
OracleBFile Class

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 1-9
- OracleBFile Class
- OracleBFile Members

OracleBFile(OracleConnection)

This constructor creates an instance of the OracleBFile class with an OracleConnection object.

Declaration

// C#
public OracleBFile(OracleConnection con);

Parameters

- con
  The OracleConnection object.

Exceptions

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

The connection must be opened explicitly by the application. OracleBFile does not open the connection implicitly.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleBFile Class
- OracleBFile Members
OracleBFile Static Fields

OracleBFile static fields are listed in Table 13–6.

Table 13–6  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>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to the value of an OracleBFile instance</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBFile Class
- OracleBFile Members

MaxSize

This static field holds the maximum number of bytes a BFILE can hold, which is 4,294,967,295 (2^32 - 1) bytes.

Declaration

```csharp
// C#
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field is useful in code that checks whether or not the operation exceeds the maximum length allowed.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBFile Class
- OracleBFile Members

Null

This static field represents a null value that can be assigned to the value of an OracleBFile instance.

Declaration

```csharp
// C#
public static readonly OracleBFile Null;
```
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBFile Class
- OracleBFile Members
OracleBFile Static Methods

OracleBFile static methods are listed in Table 13–7.

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBFile Class
- OracleBFile Members
OracleBFile Instance Properties

OracleBFile instance properties are listed in Table 13–8.

Table 13–8 OracleBFile Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether or not the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether or not forward and backward seek operations can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether or not 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>IsNull</td>
<td>Indicates whether or not the current instance has a null value</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 1-9
- OracleBFile Class
- OracleBFile Members

CanRead

Overrides Stream

This instance property indicates whether or not 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 1-9
- OracleBFile Class
- OracleBFile Members
CanSeek

Overrides Stream

This instance property indicates whether or not 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 1-9
- OracleBFile Class
- OracleBFile Members

CanWrite

Overrides Stream

This instance property indicates whether or not 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 1-9
- 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`.
OracleBFile Instance Properties

Exceptions
ObjectDisposedException - The object is already disposed.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBFile Class
- OracleBFile Members

DirectoryName
This instance property indicates the directory alias of the BFILE.

Declaration
// C#
public string DirectoryName {get;set;}

Property Value
A string.

Exceptions
ObjectDisposedException - The object is already disposed.
InvalidOperationException - The value of the DirectoryName changed while the BFILE is open.

Remarks
The maximum length of a DirectoryName is 30 bytes.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- 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.
IsNull

This property indicates whether or not 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 1-9
- 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 1-9
- OracleBFile Class
- OracleBFile Members

Length

Overides 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 1-9
- OracleBFile Class
- OracleBFile Members

Position

Overrides Stream

This instance property indicates the current read position in the LOB stream.

Declaration

// C#
public override Int64 Position{get; set;}

Property Value

An Int64 value that indicates the read position.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The value is less than 0.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleBFile Class
- OracleBFile Members
OracleBFile Instance Methods

OracleBFile instance methods are listed in Table 13–9.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Inherited from System.IO.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 System.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 System.IO.Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td>Not Supported</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.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 System.IO.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 System.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

// C#
public object Clone();

Return Value

An OracleBFile object.

Implements

ICloneable

Exceptions

- ObjectDisposedException - The object is already disposed.
- InvalidOperation Exception - 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

// Database Setup, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has 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 previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // 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();

ccon.Close();
ccon.Dispose();
}
}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleBFile Class
- OracleBFile Members

CloseFile

This instance method closes the BFILE referenced by the current BFILE instance.
OracleBFile Instance Methods

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 1-9
- 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, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has 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 previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // Note that the byte representation of the ABCDABC is 656667686667
        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();
    }
}
OracleBFile Instance Methods

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleBFile Class
- OracleBFile 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.

See Also:
■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ 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:
OracleBFile Instance Methods

- 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 1-9
- OracleBFile Class
- OracleBFile Members

CopyTo(Int64, OracleBlob, Int64, Int64)
This instance method copies data from the current OracleBFile instance to the provided OracleBlob object with the specified source offset, destination offset, and character amounts.

Declaration

// C#
public Int64 CopyTo(Int64 src_offset, OracleBlob obj, Int64 dst_offset, Int64 amount);

Parameters
- src_offset
  The offset (in bytes) in the current instance, from which the data is read.
- obj
  An OracleBlob object to which the data is copied.
- dst_offset
  The offset (in bytes) to which the OracleBlob object is copied.
- amount
  The amount of data to be copied.

Return Value
The return value is the amount copied.

Exceptions
ObjectDisposedException - The object is already disposed.
ArgumentOutOfRangeException - The src_offset, the dst_offset, or the amount is less than 0.
InvalidOperationException - This exception is thrown if any of the following conditions exist:
OracleBFile Class

- 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 1-9
- **OracleBFile Class**
- **OracleBFile Members**

**CopyTo(OracleClob)**
This instance method copies data from the current **OracleBFile** instance to the provided **OracleClob** object.

**Declaration**
```
// C#
public Int64 CopyTo(OracleClob obj);
```

**Parameters**
- **obj**
  - The **OracleClob** object to which the data is copied.

**Return Value**
The return value is the amount copied.

**Exceptions**
- **ObjectDisposedException** - The object is already disposed.
- **InvalidOperationException** - This exception is thrown if any of the following conditions exist:
  - The **OracleConnection** is not open or has been closed during the lifetime of the object.
  - The LOB object parameter has a different connection than the object.

**Remarks**
The provided object and the current instance must be using the same connection, that is, the same **OracleConnection** object.
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 1-9
- 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 1-9
- OracleBFile Class
- OracleBFile Members
Dispose

This instance method releases resources allocated by this object.

**Declaration**

```csharp
// C#
public void Dispose();
```

**Implements**

IDisposable

**Remarks**

Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls may lead to exceptions. The object cannot be reused after being disposed.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBFile Class
- OracleBFile Members

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.
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 1-9
- 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, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has CREATE ANY DIRECTORY privilege.
CREATE OR REPLACE DIRECTORY MYDIR AS 'C:\TEMP';
*/

using System;
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;
class ReadSample
{
static void Main()
{
   // Create MYDIR directory object as indicated previously and create a file
   // MyFile.txt with the text ABCDABC under C:\TEMP directory.
   // 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] + ");
   }
}
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, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has 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 previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // Note that the byte representation of the ABCDABC is 656667686667

        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();
    }
}
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, if you have not done so yet.
/* Log on as DBA (SYS or SYSTEM) that has 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 previously and create a file
        // MyFile.txt with the text ABCDABC under C:\TEMP directory.
        // 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"
        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 1-9
- 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
System.Object
    System.MarshalByRefObject
        System.IO.Stream
            Oracle.DataAccess.Types.OracleBlob

Declaration
// C#
public sealed class OracleBlob : Stream, ICloneable, INullable

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();
}

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Members
- OracleBlob Constructors
- OracleBlob Static Fields
- OracleBlob Static Methods
- OracleBlob Instance Properties
- OracleBlob Instance Methods
OracleBlob Members

OracleBlob members are listed in the following tables.

OracleBlob Constructors
OracleBlob constructors are listed in Table 13–10.

Table 13–10  OracleBlob Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleBlob Constructors</td>
<td>Creates an instance of the OracleBlob class (Overloaded)</td>
</tr>
</tbody>
</table>

OracleBlob Static Fields
OracleBlob static fields are listed in Table 13–11.

Table 13–11  OracleBlob 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 BLOB can hold, which is 4,294,967,295 (2^32 - 1) bytes</td>
</tr>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to the value of an OracleBlob instance</td>
</tr>
</tbody>
</table>

OracleBlob Static Methods
OracleBlob static methods are listed in Table 13–12.

Table 13–12  OracleBlob Static Methods

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

OracleBlob Instance Properties
OracleBlob instance properties are listed in Table 13–13.

Table 13–13  OracleBlob Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether or not the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether or not forward and backward seek operations be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether or not 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 or not the BLOB has been opened to defer index updates</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the current instance has a null value</td>
</tr>
</tbody>
</table>
OracleBlob Class

OracleBlob Instance Methods
OracleBlob instance methods are listed in Table 13–14.

Table 13–14  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 System.IO.Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from System.IO.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 System.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 System.IO.Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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 System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializedLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
</tbody>
</table>

Table 13–13 (Cont.)  OracleBlob Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance</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>
**Table 13–14 (Cont.) OracleBlob Instance Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 System.IO.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 System.Object</td>
</tr>
<tr>
<td>Write</td>
<td>Writes the supplied buffer into the OracleBlob</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Inherited from System.IO.Stream</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Members
OracleBlob Constructors

OracleBlob constructors are listed in Table 13–10.

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 1-9
■ OracleBlob Class
■ OracleBlob Members

OracleBlob(OracleConnection)

This constructor creates an instance of the OracleBlob class bound to a temporary BLOB with an OracleConnection object.

Declaration

// C#
public OracleBlob(OracleConnection con);

Parameters

■ con

The OracleConnection object.

Exceptions

InvalidOperationException - The OracleConnection is not opened.

Remarks

The connection must be opened explicitly by the application. OracleBlob does not open the connection implicitly.

The temporary BLOB utilizes the provided connection to store BLOB data. Caching is not turned on by this constructor.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ OracleBlob Class
■ OracleBlob Members
OracleBlob(OracleConnection, bool)

This constructor creates an instance of the OracleBlob class bound to a temporary BLOB with an OracleConnection object and a boolean value for caching.

Declaration

// C#
public OracleBlob(OracleConnection con, bool bCaching);

Parameters

■ con
The OracleConnection object.

■ bCaching
A flag for enabling or disabling server-side caching.

Exceptions

InvalidOperationException - The OracleConnection is not opened.

Remarks

The connection must be opened explicitly by the application. OracleBlob does not open the connection implicitly.

The temporary BLOB uses the provided connection to store BLOB data. The bCaching input parameter determines whether or not server-side caching is used.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9

■ OracleBlob Class

■ OracleBlob Members
OracleBlob Static Fields

OracleBlob static fields are listed in Table 13–15.

<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>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to the value of an OracleBlob instance</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 or not the operation exceeds the maximum length allowed.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

Null

This static field represents a null value that can be assigned to the value of an OracleBlob instance.

Declaration

```csharp
// C#
public static readonly OracleBlob Null;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members
OracleBlob Static Methods

OracleBlob static methods are listed in Table 13–16.

Table 13–16  OracleBlob Static Methods

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members
OracleBlob Instance Properties

OracleBlob instance properties are listed in Table 13–17.

Table 13–17 OracleBlob Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether or not the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether or not forward and backward seek operations be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether or not 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 or not the BLOB has been opened to defer index updates</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the current instance has a null value</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 optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance</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 1-9
- OracleBlob Class
- OracleBlob Members

CanRead

Overrides Stream
This instance property indicates whether or not the LOB stream can be read.

Declaration

// C#
public override bool CanRead();

Property Value
If the LOB stream can be read, returns true; otherwise, returns false.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members
OracleBlob Instance Properties

CanSeek

Overrides Stream

This instance property indicates whether or not 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 1-9
- OracleBlob Class
- OracleBlob Members

CanWrite

Overrides Stream

This instance property indicates whether or not 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 1-9
- OracleBlob Class
- OracleBlob Members

Connection

This instance property indicates the OracleConnection that is used to retrieve and write BLOB data.

Declaration

// C#
public OracleConnection Connection { get; }

Property Value

An object of OracleConnection.

Exceptions

ObjectDisposedException - The object is already disposed.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

IsEmpty

This instance property indicates whether the BLOB is empty or not.

**Declaration**

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

**Property Value**

A bool that indicates whether or not the BLOB is empty.

**Exceptions**

ObjectDisposedException - The object is already disposed.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

IsInChunkWriteMode

This instance property indicates whether or not the BLOB has been opened to defer index updates.

**Declaration**

```csharp
// 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 1-9
- OracleBlob Class
- OracleBlob Members

IsNull

This property indicates whether or not the current instance has a null value.

**Declaration**

```csharp
// C#
public bool IsNull {get;}
```
OracleBlob Instance Properties

**Property Value**
Returns true if the current instance has a null value; otherwise, returns false.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

**IsTemporary**
This instance property indicates whether or not the current instance is bound to a temporary BLOB.

**Declaration**

```csharp
public bool IsTemporary {get;}
```

**Property Value**
bool

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

**Length**
Overides Stream
This instance property indicates the size of the BLOB data in bytes.

**Declaration**

```csharp
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 1-9
- OracleBlob Class
- OracleBlob Members
OptimumChunkSize

This instance property indicates the optimal data buffer length (or multiples thereof) that read and write operations should use to improve performance.

Declaration

```csharp
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 1-9
- `OracleBlob Class`
- `OracleBlob Members`

Position

Overrides `Stream`

This instance property indicates the current read or write position in the LOB stream.

Declaration

```csharp
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 1-9
- `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
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 1-9
- OracleBlob Class
- OracleBlob Members
OracleBlob Instance Methods

OracleBlob instance methods are listed in Table 13–18.

<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 System.IO.Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from System.IO.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 System.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 System.IO.Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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 System.Object</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializedLifetimeService</td>
<td>Inherited from System.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 System.IO.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 System.Object</td>
</tr>
<tr>
<td>Write</td>
<td>Writes the supplied buffer into the OracleBlob</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Inherited from System.IO.Stream</td>
</tr>
</tbody>
</table>
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

Append

Append appends the supplied data to the end of the current OracleBlob instance.

Overload List:
- Append(OracleBlob)
  This instance method appends the BLOB data referenced by the provided OracleBlob object to the current OracleBlob instance.
- Append(byte[], int, int)
  This instance method appends data from the supplied byte array buffer to the end of the current OracleBlob instance.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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.
**OracleBlob Class**

---

**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 AppendSample
{
    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);

        // Append 2 bytes {4, 5} to the OracleBlob
        byte[] buffer = new byte[3] {4, 5, 6};
        blob.Append(buffer, 0, 2);

        byte[] appendBuffer = blob.Value;

        // Prints 'appendBuffer = 45'
        Console.Write("appendBuffer = ");
    }
}
```

---

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members
for(int index = 0; index < appendBuffer.Length; index++)
{
    Console.Write(appendBuffer[index]);
} Console.WriteLine();

blob.Close();
blob.Dispose();

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

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

BeginChunkWrite

This instance method opens the BLOB.

Declaration

// C#
public void BeginChunkWrite();

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks

BeginChunkWrite does not need to be called before manipulating the BLOB data. This is provided for performance reasons.

After this method is called, write operations do not cause the domain or function-based index on the column to be updated. Index updates occur only once after EndChunkWrite is called.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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

```csharp
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();

        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();
    }
}
```
Close

This instance method closes the current stream and releases any resources associated with it.

Declaration

```csharp
// C#
public override void Close();
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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.

InvalidOperationException - The parameter has a different connection than the object, OracleConnection is not opened, or OracleConnection has been reopened.

ArgumentOutOfRangeException - The src_offset, the dst_offset, or the amount parameter is less than 0.

Remarks

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
  - OracleBlob Class
  - OracleBlob Members

CopyTo(OracleBlob)

This instance method copies data from the current instance to the provided OracleBlob object.

Declaration

// C#
public Int64 CopyTo(OracleBlob obj);

Parameters

- obj
The `OracleBlob` object to which the data is copied.

**Return Value**
The return value is the amount copied.

**Exceptions**
- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - This exception is thrown if any of the following conditions exist:
  - The `OracleConnection` is not open or has been closed during the lifetime of the object.
  - The LOB object parameter has a different connection than the object.

**Remarks**
The provided object and the current instance must be using the same connection; that is, the same `OracleConnection` object.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

**CopyTo(OracleBlob, Int64)**
This instance method copies data from the current `OracleBlob` instance to the provided `OracleBlob` object with the specified destination offset.

**Declaration**

```csharp
// 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 1-9
- OracleBlob Class
- OracleBlob Members

**CopyTo(Int64, OracleBlob, Int64, Int64)**

This instance method copies data from the current OracleBlob instance to the provided OracleBlob object with the specified source offset, destination offset, and character amounts.

**Declaration**

// C#
public Int64 CopyTo(Int64 `src_offset`, OracleBlob `obj`, Int64 `dst_offset`,
                   Int64 `amount`);

**Parameters**

- `src_offset`
  The offset (in bytes) in the current instance, from which the data is read.

- `obj`
  The OracleBlob object to which the data is copied.

- `dst_offset`
  The offset (in bytes) at which the OracleBlob object is copied.

- `amount`
  The amount of data to be copied.

**Return Value**

The return value is the amount copied.

**Exceptions**

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The parameter has a different connection than the object, `OracleConnection` is not opened, or `OracleConnection` has been reopened.

ArgumentException - The `src_offset`, the `dst_offset`, or the `amount` parameter is less than 0.
Remarks
If the dst_offset is beyond the end of the OracleBlob data, spaces are written into the OracleBlob until the dst_offset is met.

The offsets are 0-based. No character conversion is performed by this operation.

The provided object and the current instance must be using the same connection; that is, the same OracleConnection object.

Example

```
// C#

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();
    }
}
```
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 1-9
- 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 1-9
- 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 1-9

■ OracleBlob Class

■ OracleBlob Members

Erase()

This instance method erases all data.

Declaration

// C#
public Int64 Erase();

Return Value

The number of bytes erased.

Remarks

Erase() replaces all data with zero-byte fillers.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9

■ OracleBlob Class

■ OracleBlob Members

Erase(Int64, Int64)

This instance method erases a specified portion of data.

Declaration

// C#
public Int64 Erase(Int64 offset, Int64 amount);

Parameters

■ **offset**
  
  The offset from which to erase.

■ **amount**
  
  The quantity (in bytes) to erase.
Return Value
The number of bytes erased.

Exceptions
ObjectDisposedException - The object is already disposed.
InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.
ArgumentOutOfRangeException - The offset or amount parameter is less than 0.

Remarks
Replaces the specified amount of data with zero-byte fillers.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

IsEqual
This instance method compares the LOB data referenced by the two OracleBlobs.

Declaration
// C#
public bool IsEqual(OracleBlob obj);

Parameters
- obj
  An OracleBlob object.

Return Value
If the current OracleBlob and the provided OracleBlob refer to the same LOB, returns true. Returns false otherwise.

Exceptions
ObjectDisposedException - The object is already disposed.
InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

Remarks
Note that this method can return true even if the two OracleBlob objects return false for == or Equals() because two different OracleBlob instances can refer to the same LOB.

The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.
OracleBlob Instance Methods

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

Read

Overrides Stream

This instance method reads a specified amount of bytes from the ODP.NET LOB instance and populates the buffer.

Declaration

```csharp
// 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

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

class ReadSample
```

---

13-66 Oracle Data Provider for .NET Developer's Guide
OracleBlob Class

{ 
    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]);
        }
        Console.WriteLine();

        blob.Close();
        blob.Dispose();

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

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBlob Class
- OracleBlob Members

Search

This instance method searches for a binary pattern in the current instance of an OracleBlob.

Declaration

// C#
public Int64 Search(byte[] val, int64 offset, int64 nth);

Parameters

- val
  
  The binary pattern being searched for.
- **offset**
  The 0-based offset (in bytes) starting from which the `OracleBlob` is searched.

- **nth**
  The specific occurrence (1-based) of the match for which the absolute offset (in bytes) is returned.

**Return Value**

Returns the absolute offset of the start of the matched pattern (in bytes) for the nth occurrence of the match. Otherwise, 0 is returned.

**Exceptions**

- **ObjectDisposedException** - The object is already disposed.
- **InvalidOperationException** - The `OracleConnection` is not open or has been closed during the lifetime of the object.
- **ArgumentOutOfRangeException** - This exception is thrown if any of the following conditions exist:
  - The **offset** is less than 0.
  - The **nth** is less than or equal to 0.
  - The **val.Length** is greater than 16383.
  - The **nth** is greater than or equal to `OracleBlob.MaxSize`.
  - The **offset** is greater than or equal to `OracleBlob.MaxSize`.

**Remarks**

The limit of the search pattern is 16383 bytes.

**Example**

```csharp
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 1-9
- OracleBlob Class
- OracleBlob Members

Seek

Overrrides 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 Int64 for the position.

Exceptions
ObjectDisposedException - The object is already disposed.
InvalidOperationExcepton - 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 1-9
- OracleBlob Class
- OracleBlob Members

SetLength

Overrides Stream

This instance method trims or truncates the BLOB value to the specified length (in bytes).

Declaration

// C#
public override void SetLength(Int64 newlen);

Parameters

- newlen
  The desired length of the current stream in bytes.

Exceptions

- ObjectDisposedException - The object is already disposed.
- InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.
- ArgumentOutOfRangeException - The newlen parameter is less than 0.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
{
    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 1-9
- 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
System.Object
  System.MarshalByRefObject
    System.IO.Stream
      Oracle.DataAccess.Types.OracleClob

Declaration
// C#
public sealed class OracleClob : Stream, ICloneable, INullable

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 13–19.

Table 13–19  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 13–20.

Table 13–20  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>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to the value of an OracleClob instance</td>
</tr>
</tbody>
</table>

OracleClob Static Methods
OracleClob static methods are listed in Table 13–21.

Table 13–21  OracleClob Static Methods

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

OracleClob Instance Properties
OracleClob instance properties are listed in Table 13–22.

Table 13–22  OracleClob Instance Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether or not the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether or not forward and backward seek operations can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether or not 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 or not the OracleClob object represents an NCLOB.</td>
</tr>
</tbody>
</table>
**OracleClob Instance Methods**

The OracleClob instance methods are listed in Table 13–23.

<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 System.IO.Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from System.IO.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 System.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 System.IO.Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>Erase</td>
<td>Erases the specified amount of data (Overloaded)</td>
</tr>
<tr>
<td>Flush</td>
<td><em>Not supported</em></td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the current instance</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.MarshalByRefObject</td>
</tr>
<tr>
<td>Methods</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</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 System.IO.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 System.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 System.IO.Stream</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
  - OracleClob Class
  - OracleClob Members

OracleClob(OracleConnection)

This constructor creates an instance of the OracleClob class bound to a temporary CLOB with an OracleConnection object.

**Declaration**

```c#
public OracleClob(OracleConnection con);
```

**Parameters**

- **con**

  The OracleConnection object.

**Exceptions**

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

**Remarks**

The connection must be opened explicitly by the application. OracleClob does not open the connection implicitly. The temporary CLOB utilizes the provided connection to store CLOB data. Caching is not enabled by default.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleClob Class
- OracleClob Members
OracleClob Static Fields

OracleClob static fields are listed in Table 13–24.

<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>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to the value of an OracleClob instance</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members

MaxSize

The MaxSize field holds the maximum number of bytes a CLOB can hold, which is 4,294,967,295 (2^32 - 1) bytes.

Declaration

```
// C#
public static readonly Int64 MaxSize = 4294967295;
```

Remarks

This field is useful in code that checks whether or not your operation exceeds the maximum length (in bytes) allowed.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members

Null

This static field represents a null value that can be assigned to the value of an OracleClob instance.

Declaration

```
// C#
public static readonly OracleClob Null;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members
OracleClob Static Methods

OracleClob static methods are listed in Table 13–25.

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members
OracleClob Instance Properties

OracleClob instance properties are listed in Table 13–26.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Indicates whether or not the LOB stream can be read</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Indicates whether or not forward and backward seek operations can be performed</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Indicates whether or not 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 or not the OracleClob object represents an NCLOB.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the current instance has a null value</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 database 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 1-9
- OracleClob Class
- OracleClob Members

CanRead

Overrides Stream
This instance property indicates whether or not 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.
OracleClob Class

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members

CanSeek

Overrides Stream
This instance property indicates whether or not 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 1-9
- OracleClob Class
- OracleClob Members

CanWrite

Overrides Stream
This instance property indicates whether or not 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 1-9
- 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;}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members
Property Value
An OracleConnection.

Exceptions
ObjectDisposedException - The object is already disposed.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleClob Class
- OracleClob Members

IsInChunkWriteMode
This instance property indicates whether or not 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 1-9
- OracleClob Class
- OracleClob Members

IsNCLOB
This instance property indicates whether or not the OracleClob object represents an NCLOB.
Declaration

// C#
public bool IsNCLOB {get;}

Property Value
A bool.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members

IsNull

This property indicates whether or not 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 1-9
- OracleClob Class
- OracleClob Members

IsTemporary

This instance property indicates whether or not the current instance is bound to a temporary CLOB.

Declaration

// C#
public bool IsTemporary {get;}

Property Value
A bool.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members

Length

Overrides Stream

This instance property indicates the size of the CLOB data in bytes.
OracleClob Instance Properties

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 1-9
- OracleClob Class
- OracleClob Members

OptimumChunkSize

This instance property indicates the minimum number of bytes to retrieve or send from the database 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 1-9
- 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 1-9
- OracleClob Class
- OracleClob Members

Value

This instance property returns the data, starting from the first character in the CLOB or NCLOB, as a string.

Declaration

// C#
public string Value{get;}

Property Value

A string.

Exceptions

ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - The Value is less than 0.

Remarks

The value of Position is neither used nor changed by using this property.

The maximum string length that can be returned by this property is 2 GB.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleClob Class
- OracleClob Members
The OracleClob instance methods are listed in Table 13–27.

**Table 13–27  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 System.IO.Stream</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Inherited from System.IO.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 System.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 System.IO.Stream</td>
</tr>
<tr>
<td>EndWrite</td>
<td>Inherited from System.IO.Stream</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.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 System.MarshalByRefObject</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Inherited from System.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 System.IO.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 System.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 System.IO.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 1-9
  - 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

```csharp
// 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 1-9
- 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 1-9
- 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();
    }
}
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 1-9
- 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();

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

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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
// C#
public Int64 CopyTo(Int64 src_offset, OracleClob obj, Int64 dst_offset,
                     Int64 amount);
```

**Parameters**

- `src_offset`
  The offset (in characters) in the current instance, from which the data is read.

- `obj`
  The OracleClob object to which the data is copied.

- `dst_offset`
  The offset (in characters) at which the OracleClob object is copied.

- `amount`
  The amount of data to be copied.

**Return Value**

The return value is the amount copied.

**Exceptions**

- `ObjectDisposedException` - The object is already disposed.
- `InvalidOperationException` - The parameter has a different connection than the object, OracleConnection is not opened, or OracleConnection has been reopened.
- `ArgumentOutOfRangeException` - The `src_offset`, the `dst_offset`, or the `amount` parameter is less than 0.

**Remarks**

If the `dst_offset` is beyond the end of the OracleClob data, spaces are written into the OracleClob until the `dst_offset` is met.

The offsets are 0-based. No character conversion is performed by this operation.
The provided object and the current instance must be using the same connection, that is, the same OracleConnection object.
Example

```csharp
// 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();
    }
}
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
  - OracleClob Class
  - OracleClob Members

### Read(byte[], int, int)

Overrides `Stream`  
This instance method reads a specified amount of bytes from the current instance and populates the byte array `buffer`.

#### Declaration

```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.
OracleClob Class

Read(char [ ], int, int)
This instance method reads a specified amount of characters from the current instance and populates the character array buffer.

Declaration
// C#
public int Read(char[] buffer, int offset, int count);

Parameters
■ buffer
  The character array buffer that is populated.
■ offset
  The offset (in characters) at which the buffer is populated.
■ count
  The amount of characters to be read.

Return Value
The return value indicates the number of characters read from the CLOB.

Exceptions
ObjectDisposedException - The object is already disposed.
InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.
ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:
■ The offset or the count is less than 0.
■ The offset is greater than or equal to the buffer.Length.
■ The offset and the count together are greater than buffer.Length.

Remarks
Handles all CLOB and NCLOB data as Unicode.
The LOB data is read starting from the position specified by the Position property.

Example
// C#
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();

    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 1-9
- 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.
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

// 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.

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();

        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 1-9
■ 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 1-9
- 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.
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 1-9
- 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.
OracleClob Instance Methods

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 1-9
- OracleClob Class
- OracleClob Members

Write(char[], int, int)
This instance method writes data from the provided character array buffer into the OracleClob.

Declaration
// C#
public void Write(char[] buffer, int offset, int count);

Parameters
- buffer
  The character array buffer that is written to the OracleClob.
- offset
  The offset (in characters) from which the buffer is read.
- count
  The amount (in characters) from the buffer that is to be written into the OracleClob.

Exceptions
ObjectDisposedException - The object is already disposed.

InvalidOperationException - The OracleConnection is not open or has been closed during the lifetime of the object.

ArgumentOutOfRangeException - This exception is thrown if any of the following conditions exist:

- The offset or the count is less than 0.
- The offset is greater than or equal to the buffer.Length.
The offset and the count together are greater than buffer.Length.

The Position is not even.

Remarks
Handles all CLOB and NCLOB data as Unicode.
The LOB data is read starting from the position specified by the Position property.
If necessary, proper data conversion is carried out from the client character set to the database character set.

Example

// C#

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 1-9
- OracleClob Class
- OracleClob Members
OracleRefCursor Class

An OracleRefCursor object represents an Oracle REF CURSOR.

Class Inheritance
System.Object
    System.MarshalByObject
        Oracle.DataAccess.Types.OracleRefCursor

Declaration
// C#
public sealed class OracleRefCursor : MarshalByObject, IDisposable, INullable

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

Remarks
To minimize the number of open server cursors, OracleRefReader objects should be explicitly disposed.

Example
// Database Setup
/*
connect scott/tiger@oracle
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;
/*

// C#

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 previously

        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;

// 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();

cn.Close();
con.Dispose();
}
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 13–28.

Table 13–28  OracleRefCursor Static Methods

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

OracleRefCursor Properties
OracleRefCursor properties are listed in Table 13–29.

Table 13–29  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>
<tr>
<td>FetchSize</td>
<td>Specifies the size that the OracleDataReader internal cache needs to store result set data</td>
</tr>
<tr>
<td>RowSize</td>
<td>Specifies the amount of memory the OracleRefcursor internal cache needs to store one row of data</td>
</tr>
</tbody>
</table>

OracleRefCursor Instance Methods
OracleRefCursor instance methods are listed in Table 13–30.

Table 13–30  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 System.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 System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRefCursor Class
OracleRefCursor Static Methods

OracleRefCursor static methods are listed in Table 13–31.

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRefCursor Class
- OracleRefCursor Members
OracleRefCursor Properties

OracleRefCursor properties are listed in Table 13–32.

<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>
<tr>
<td>FetchSize</td>
<td>Specifies the size that the OracleDataReader internal cache needs to</td>
</tr>
<tr>
<td></td>
<td>store result set data</td>
</tr>
<tr>
<td>RowSize</td>
<td>Specifies the amount of memory the OracleRefcursor internal cache needs to</td>
</tr>
<tr>
<td></td>
<td>store one row of data</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleRefCursor Class
- OracleRefCursor Members
FetchSize

This property specifies the size that the OracleDataReader internal cache needs to store result set data.

Declaration

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

Property Value

A long that specifies the size (in bytes) of the OracleRefCursor internal cache.

Exceptions

ArgumentException - The FetchSize value specified is invalid.

Remarks

Default = 131072.

The FetchSize property value is inherited by the OracleCommand that created the OracleRefCursor object. The FetchSize property on the OracleDataReader object determines the amount of data the OracleRefCursor fetches into its internal cache for each database round-trip.

This property is useful if the OracleRefCursor is explicitly used to fill the DataSet or DataTable through the OracleDataAdapter, because it can provide control on how the data of the REF CURSOR is fetched.

If an OracleDataReader object is created from the OracleRefCursor, the resulting OracleDataReader object inherits the FetchSize value of the OracleDataReader object. However, the inherited value can be overridden, if it is set before the first invocation of the OracleDataReader Read method for the given result set, by setting the OracleDataReader FetchSize property.

The RowSize and FetchSize properties handle UDT and XMLType data differently than other scalar data types. Because only a reference to the UDT and XMLType data is stored in the ODP.NET's internal cache, the RowSize property accounts for only the memory needed for the reference (which is very small) and not the actual size of the UDT and XMLType data. Thus, applications can inadvertently fetch a large number of UDT or XMLType instances from the database in a single database round-trip. This is because the actual size of UDT and XMLType data does not count against the FetchSize, and it would require numerous UDT and XMLType references to fill up the default cache size of 131072 bytes. Therefore, when fetching UDT or XMLType data, the FetchSize property must be appropriately configured to control the number of UDT and XMLType instances that are to be fetched, rather than the amount of the actual UDT and XMLType data to be fetched.

NOTE: For LOB and LONG data types, only the sizes specified in the InitialLOBFetchSize and InitialLONGFetchSize properties are accounted for by the RowSize property in addition to the metadata and reference information that is maintained by the cache for each LOB in the select list.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRefCursor Class
- OracleRefCursor Members
RowSize
This property specifies the amount of memory the OracleRefcursor internal cache needs 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 OracleRefcursor needs to store one row of data for the executed query.

Remarks
The RowSize property is set to a nonzero value when the OracleRefcursor object is created. 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 OracleRefcursor to fetch $N$ rows for each database round-trip, the OracleRefcursor FetchSize property can be set dynamically to $\text{RowSize} \times N$. Note that for the FetchSize to take effect appropriately, it must be set before the it is used to fill the DataSet/DataTable using OracleDataAdapter.

If an OracleDataReader is obtained from the OracleRefCursor through the GetDataReader method, the resulting OracleDataReader will have its FetchSize property set to the FetchSize value of the OracleRefCursor.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRefCursor Class
- OracleRefCursor Members
OracleRefCursor Instance Methods

OracleRefCursor instance methods are listed in Table 13–33.

<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 System.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 System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>

**Dispose**

This instance method disposes of the resources allocated by the OracleRefCursor object.

**Declaration**

// C#
public void Dispose();

**Implements**

IDisposable

**Remarks**

The object cannot be reused after being disposed.

Once Dispose() is called, the object of OracleRefCursor is in an uninitialized state. Although some properties can still be accessed, their values may not be accountable. Since resources are freed, method calls can lead to exceptions.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRefCursor Class
- OracleRefCursor Members

**GetDataReader**

This instance method returns an OracleDataReader object for the REF CURSOR.

**Declaration**

// C#
public OracleDataReader GetDataReader();

Return Value
OracleDataReader

Remarks
Using the OracleDataReader, rows can be fetched from the REF CURSOR.

See Also:
■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ OracleRefCursor Class
■ OracleRefCursor Members
This chapter describes the ODP.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
- INullable Interface
OracleBinary Structure

The OracleBinary structure represents a variable-length stream of binary data to be stored in or retrieved from a database.

Class Inheritance

System.Object
  System.ValueType
    Oracle.DataAccess.Types.OracleBinary

Declaration

// ADO.NET 2.0: C#
public struct OracleBinary : IComparable, INullable, IXmlSerializable

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 14–1

Table 14–1  OracleBinary Constructors

<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 14–2.

Table 14–2  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>

OracleBinary Static Methods
The OracleBinary static methods are listed in Table 14–3.

Table 14–3  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>

OracleBinary Static Operators
The OracleBinary static operators are listed in Table 14–4.

Table 14–4  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>
</tbody>
</table>
Oracle Binary Structure

Oracle Binary Static Type Conversion Operators

The OracleBinary static type conversion operators are listed in Table 14–5.

Table 14–5  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>

Oracle Binary Properties

The OracleBinary properties are listed in Table 14–6.

Table 14–6  OracleBinary Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNull</td>
<td>Indicates whether or not 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>

Oracle Binary Instance Methods

The OracleBinary instance methods are listed in Table 14–7.

Table 14–7  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 System.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 1-9
- 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

```csharp
// C#
public OracleBinary(byte[] bytes);
```

Parameters

- `bytes`  
  A byte array.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBinary Structure
- OracleBinary Members
OracleBinary Static Fields

The OracleBinary static fields are listed in Table 14–8.

<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 1-9
- 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 1-9
- OracleBinary Structure
- OracleBinary Members
OracleBinary Static Methods

The OracleBinary static methods are listed in Table 14–9.

Table 14–9  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 1-9
- OracleBinary Structure
- OracleBinary Members

Concat

This method returns the concatenation of two OracleBinary structures.

Declaration

// C#
public static OracleBinary Concat(OracleBinary value1, OracleBinary value2);

Parameters

- value1
  - The first OracleBinary.
- value2
  - The second OracleBinary.

Return Value

An OracleBinary.

Remarks

If either argument has a null value, the returned OracleBinary structure has a null value.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBinary Structure
- OracleBinary Members

Equals

This method determines if two OracleBinary values are equal.

Declaration

// C#
public static bool Equals(OracleBinary value1, OracleBinary value2);

Parameters

- value1
  The first OracleBinary.
- value2
  The 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 1-9
- OracleBinary Structure
- OracleBinary Members

GreaterThanOrEqual

This method determines whether or not the first of two OracleBinary values is greater than the second.

Declaration

// C#
public static bool GreaterThan(OracleBinary value1, OracleBinary value2);

Parameters

- value1
  The first OracleBinary.
- value2
  The second OracleBinary.
Return Value
Returns true if the first of two OracleBinary values is greater than the second; otherwise returns false.

Remarks
The following rules apply to the behavior of this method.

■ Any OracleBinary that has a value is greater than an OracleBinary that has a null value.
■ Two OracleBinaries 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 1-9
■ OracleBinary Structure
■ OracleBinary Members

GreaterThanOrEqual
This method determines whether or not 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
    The first OracleBinary.
■ value2
    The 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 OracleBinarys that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBinary Structure
- OracleBinary Members

LessThan

This method determines whether or not the first of two OracleBinary values is less than the second.

Declaration

// C#
public static bool LessThan(OracleBinary value1, OracleBinary value2);

Parameters

- value1
  The first OracleBinary.
- value2
  The 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 OracleBinarys that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBinary Structure
- OracleBinary Members
LessThanOrEqual

This method determines whether or not 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`  
  The first `OracleBinary`.
- `value2`  
  The 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 `OracleBinarys` that contain a null value are equal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleBinary Structure`
- `OracleBinary Members`

NotEquals

This method determines whether or not two `OracleBinary` values are not equal.

**Declaration**

```csharp
// C#
public static bool NotEquals(OracleBinary value1, OracleBinary value2);
```

**Parameters**

- `value1`  
  The first `OracleBinary`.
- `value2`  
  The second `OracleBinary`.

**Return Value**

Returns `true` if two `OracleBinary` values are not equal; otherwise returns `false`.

**Remarks**

The following rules apply to the behavior of this method.
Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.

Two `OracleBinary`s that contain a null value are equal.

**See Also:**
- "`Oracle.DataAccess.Types Namespace`" on page 1-9
- `OracleBinary Structure`
- `OracleBinary Members`
OracleBinary Static Operators

The OracleBinary static operators are listed in Table 14–10.

Table 14–10 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 1-9
- OracleBinary Structure
- OracleBinary Members

operator +

This method concatenates two OracleBinary values.

Declaration

```csharp
// C#
public static OracleBinary operator + (OracleBinary value1, OracleBinary value2);
```

Parameters

- `value1`  
  The first OracleBinary.
- `value2`  
  The second OracleBinary.

Return Value

OracleBinary

Remarks

If either argument has a null value, the returned OracleBinary structure has a null value.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBinary Structure
- OracleBinary Members

operator ==
This method determines if two OracleBinary values are equal.

Declaration
// C#
public static bool operator == (OracleBinary value1, OracleBinary value2);

Parameters
- value1
  The first OracleBinary.
- value2
  The 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 1-9
- 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
  The first OracleBinary.
- value2
  The second OracleBinary.
Return Value

Returns true if the first of two OracleBinary values is greater than the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleBinary that has a value is greater than an OracleBinary that has a null value.
- Two OracleBinarys 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 1-9
- 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`
  - The first OracleBinary.
- `value2`
  - The 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 1-9
- `OracleBinary Structure`
- `OracleBinary Members`

**operator !=**
This method determines if two `OracleBinary` values are not equal.

**Declaration**
```c#
public static bool operator != (OracleBinary value1, OracleBinary value2);
```

**Parameters**
- `value1`  
The first `OracleBinary`.
- `value2`  
The 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 1-9
- `OracleBinary Structure`
- `OracleBinary Members`

**operator <**
This method determines if the first of two `OracleBinary` values is less than the second.

**Declaration**
```c#
public static bool operator < (OracleBinary value1, OracleBinary value2);
```

**Parameters**
- `value1`
The first OracleBinary.

- value2
  The 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 1-9
- OracleBinary Structure
- OracleBinary Members

**operator <=**

This method determines if the first of two OracleBinary values is less than or equal to the second.

**Declaration**

// C#
public static bool operator <= (OracleBinary value1, OracleBinary value2);

**Parameters**

- value1
  The first OracleBinary.
- value2
  The 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 1-9
- OracleBinary Structure
- OracleBinary Members
OracleBinary Static Type Conversion Operators

The OracleBinary static type conversion operators are listed in Table 14–11.

Table 14–11  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 1-9
- OracleBinary Structure
- OracleBinary Members

**explicit operator byte[ ]**

This method converts an OracleBinary value to a byte array.

**Declaration**

```csharp
// 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 1-9
- OracleBinary Structure
- OracleBinary Members

**implicit operator OracleBinary**

This method converts a byte array to an OracleBinary structure.

**Declaration**

```csharp
// C#
public static implicit operator OracleBinary(byte[ ] bytes);
```

**Parameters**

- `bytes`
A byte array.

**Return Value**
OracleBinary

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleBinary Structure
- OracleBinary Members
OracleBinary Properties

The OracleBinary properties are listed in Table 14–12.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNull</td>
<td>Indicates whether or not 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 1-9
- OracleBinary Structure
- OracleBinary Members

IsNull

This property indicates whether or not 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 1-9
- OracleBinary Structure
- OracleBinary Members

Item

This property obtains the particular byte in an OracleBinary structure using an index.

Declaration

```csharp
// 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

```csharp
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 1-9
- OracleBinary Structure
- OracleBinary Members

Length

This property returns the length of the binary data.

Declaration

```csharp
// C#
public int length {get;}
```

Property Value

Length of the binary data.

Exceptions

OracleNullValueException - The current instance has a null value.

Example

```csharp
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**

```csharp
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 1-9
- `OracleBinary` Structure
- `OracleBinary Members`
OracleBinary Instance Methods

The OracleBinary instance methods are listed in Table 14–13.

Table 14–13  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 System.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 1-9
- OracleBinary Structure
- OracleBinary Members

**CompareTo**

This method compares the current instance to an object and returns an integer that represents their relative values

**Declaration**

// C#
public int CompareTo(object obj);

**Parameters**

- obj
  
The object being compared.

**Return Value**

The method returns a number that is:

- Less than zero: if the current OracleBinary instance value is less than obj.
- Zero: if the current OracleBinary instance and obj values have the same binary data.
- Greater than zero: if the current OracleBinary instance value is greater than obj.

**Implements**

IComparable

**Exceptions**

ArgumentException - The parameter is not of type OracleBinary.
Remarks
The following rules apply to the behavior of this method.

- The comparison must be between OracleBinarys. 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 OracleBinarys that contain a null value are equal.

Example

```csharp
// 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 1-9
- OracleBinary Structure
- OracleBinary Members

Equals
This method determines whether or not an object is an instance of OracleBinary, and has the same binary data as the current instance.

Declaration

```csharp
// C#
public override bool Equals(object obj);
```

Parameters

- `obj`:
  The object being compared.

Return Value
Returns true if `obj` is an instance of OracleBinary, and has the same binary data as the current instance; otherwise, returns false.
Remarks
The following rules apply to the behavior of this method.

- Any `OracleBinary` that has a value is greater than an `OracleBinary` that has a null value.
- Two `OracleBinary`s that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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 1-9
- `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
`string`

Remarks
If the current `OracleBinary` instance has a null value, the returned string "null".

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleBinary` Structure
- `OracleBinary` Members
OracleDate Structure

The OracleDate structure represents the Oracle DATE data type to be stored in or retrieved from a database. Each OracleDate stores the following information: year, month, day, hour, minute, and second.

Class Inheritance
System.Object
    System.ValueType
        Oracle.DataAccess.Types.OracleDate

Declaration
// ADO.NET 2.0: C#
public struct OracleDate : IComparable, INullable, IXmlSerializable

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{0}\nto\n{1}\n", date1, date2);
        Console.WriteLine(".NET DateTime ranges from\n{0}\nto\n{1}\n", date3, date4);
    }
}

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 14–14

<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 14–15.

<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 14–16.

<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 14–17.
OracleDate Members

Table 14–17  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 14–18.

Table 14–18  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 14–19.

Table 14–19  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 or not 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 14–20.
### Table 14–20 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 or not 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 System.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 1-9
- 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**

// 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 the Oracle 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 1-9
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class” on page 10-2
- 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 1-9
- 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**

- **ArgumentNullException** - 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 1-9
- OracleDate Structure
- OracleDate Members

**OracleDate(byte [])**

This constructor creates a new instance of the OracleDate structure and sets its value to the provided byte array, which is in the internal Oracle DATE format.

**Declaration**

```csharp
// C#
public OracleDate(byte [] bytes);
```

**Parameters**

- **bytes**
  A byte array that represents Oracle DATE in the internal Oracle DATE format.

**Exceptions**

- **ArgumentNullException** - 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 1-9
- OracleDate Structure
- OracleDate Members
OracleDate Static Fields

The `OracleDate` static fields are listed in Table 14–21.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxValue</td>
<td>Represents the maximum valid date for an <code>OracleDate</code> structure, which is December 31, 9999 23:59:59</td>
</tr>
<tr>
<td>MinValue</td>
<td>Represents the minimum valid date for an <code>OracleDate</code> 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 <code>OracleDate</code> structure instance</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleDate` Structure
- `OracleDate` Members

MaxValue

This static field represents the maximum valid date for an `OracleDate` structure, which is December 31, 9999 23:59:59.

Declaration

```
// C#
public static readonly OracleDate MaxValue;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleDate` Structure
- `OracleDate` Members

MinValue

This static field represents the minimum valid date for an `OracleDate` structure, which is January 1, -4712.

Declaration

```
// C#
public static readonly OracleDate MinValue;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleDate` Structure
- `OracleDate` Members
**Null**

This static field represents a null value that can be assigned to the value of an `OracleDate` instance.

**Declaration**

```csharp
// C#
public static readonly OracleDate Null;
```

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleDate Structure`
- `OracleDate Members`
OracleDate Static Methods

The OracleDate static methods are listed in Table 14–22.

Table 14–22  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>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
  The first OracleDate.
- value2
  The 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 1-9
- OracleDate Structure
- OracleDate Members

**GreaterThanOrEqualTo**

This method determines if the first of two OracleDate values is greater than or equal to the second.

**Declaration**

// C#
public static bool GreaterThanOrEqualTo(OracleDate value1, OracleDate value2);

**Parameters**

- value1
  The first OracleDate.
- value2
  The 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 1-9
- OracleDate Structure
- OracleDate Members

**GreaterThanOrEqualToOrEqual**

This method determines if the first of two OracleDate values is greater than or equal to the second.

**Declaration**

// C#
public static bool GreaterThanOrEqualToOrEqual(OracleDate value1, OracleDate value2);
Parameters

- `value1`
  The first `OracleDate`.
- `value2`
  The 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 1-9
- `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`
  The first `OracleDate`.
- `value2`
  The 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.
OracleDate Static Methods

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDate Structure
- OracleDate Members

LessThanOrEqual

This method determines if the first of two OracleDate values is less than or equal to the second.

Declaration

```csharp
// C#
public static bool LessThanOrEqual(OracleDate value1, OracleDate value2);
```

Parameters

- **value1**
  The first OracleDate.
- **value2**
  The 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 1-9
- OracleDate Structure
- OracleDate Members

NotEquals

This method determines if two OracleDate values are not equal.

Declaration

```csharp
// C#
public static bool NotEquals(OracleDate value1, OracleDate value2);
```

Parameters

- **value1**
  The first OracleDate.
- **value2**
  The second OracleDate.
The 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 1-9
- 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 1-9
- 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 the Oracle 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 1-9
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
- Oracle Database SQL Reference for further information on datetime format elements
OracleDate Static Operators

The OracleDate static operators are listed in Table 14-23.

Table 14–23  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 1-9
- 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
  The first OracleDate.
- value2
  The 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 1-9
- OracleDate Structure
- OracleDate Members

**operator >**

This method determines if the first of two `OracleDate` values is greater than the second.

**Declaration**

```csharp
// C#
public static bool operator > (OracleDate value1, OracleDate value2);
```

**Parameters**

- `value1`  
  The first `OracleDate`.
- `value2`  
  The 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 1-9
- OracleDate Structure
- OracleDate Members

**operator >=**

This method determines if the first of two `OracleDate` values is greater than or equal to the second.

**Declaration**

```csharp
// C#
public static bool operator >= (OracleDate value1, OracleDate value2);
```

**Parameters**

- `value1`  
  The first `OracleDate`.
- `value2`  
  The second `OracleDate`.

14-48  Oracle Data Provider for .NET Developer's Guide
The 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 1-9
- OracleDate Structure
- OracleDate Members

**operator !=**
This method determines if the two OracleDate values are not equal.

**Declaration**
```csharp
// C# public static bool operator != (OracleDate value1, OracleDate value2);
```

**Parameters**
- `value1`
The first OracleDate.
- `value2`
The 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 1-9
- OracleDate Structure
- OracleDate Members

**operator <**
This method determines if the first of two OracleDate values is less than the second.

**Declaration**
```csharp
// C# public static bool operator < (OracleDate value1, OracleDate value2);
```
Declaration

// C#
public static bool operator < (OracleDate value1, OracleDate value2);

Parameters

- value1
  The first OracleDate.
- value2
  The 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 1-9
- 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
  The first OracleDate.
- value2
  The 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 1-9
- OracleDate Structure
- OracleDate Members
OracleDate Static Type Conversions

The OracleDate static type conversions are listed in Table 14–24.

**Table 14–24  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>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDate Structure
- OracleDate Members

**explicit operator DateTime**

This method converts an OracleDate structure to a DateTime structure.

**Declaration**

```csharp
// 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 1-9
- OracleDate Structure
- OracleDate Members

**explicit operator OracleDate**

explicit operator OracleDate converts the provided structure to an 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 1-9
- 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 1-9
- 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 the Oracle 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 1-9
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
OracleDate Properties

The OracleDate properties are listed in Table 14–25.

Table 14–25  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 or not 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 1-9
- 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 1-9
- 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.

IsNull
This property indicates whether or not 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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.
OracleDate Structure

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 1-9
- 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 1-9
- OracleDate Structure
- OracleDate Members
OracleDate Methods

The OracleDate methods are listed in Table 14–26.

<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 or not 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 System.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 1-9
- 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

ArgumentException - The `obj` parameter is not an instance of OracleDate.
Remarks
The following rules apply to the behavior of this method.

- The comparison must be between OracleDates. For example, comparing an OracleDate instance with an OracleBinary instance is not allowed. When an OracleDate is compared with a different type, an ArgumentException is thrown.
- Any OracleDate that has a value compares greater than an OracleDate that has a null value.
- Two OracleDates that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDate Structure
- OracleDate Members

Equals
This method determines whether or not 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 1-9
- 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 1-9
  - 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 1-9
  - 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 1-9
- 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'
```csharp
Console.WriteLine(date.ToString());
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDate Structure
- OracleDate Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
The OracleDecimal structure represents an Oracle NUMBER in the database or any Oracle numeric value.

**Class Inheritance**

System.Object
   System.ValueType
      Oracle.DataAccess.Types.OracleDecimal

**Declaration**

// ADO.NET 2.0: C#  
public struct OracleDecimal : IComparable, INullable, IXmlSerializable

**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 data type can only hold up to 28 precision. When accessing the OracleDecimal.Value property from an OracleDecimal that has a value greater than 28 precision, an exception is thrown. 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);
    }
}
OracleDecimal Structure

Requirements

Namespace: `Oracle.DataAccess.Types`

Assembly: `Oracle.DataAccess.dll`

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 14–27

Table 14–27 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 (Overloaded)</td>
</tr>
</tbody>
</table>

OracleDecimal Static Fields
The OracleDecimal static fields are listed in Table 14–28.

Table 14–28 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\ldots9 \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 (-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>

OracleDecimal Static (Comparison) Methods
The OracleDecimal static (comparison) methods are listed in Table 14–29.

Table 14–29 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 14–29 (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 14–30.

**Table 14–30 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 14–31.

Table 14–31 OracleDecimal Static (Logarithmic) Methods

<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 14–32.

Table 14–32 OracleDecimal Static (Trigonometric) Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acos</td>
<td>Returns an angle in radians whose cosine is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Asin</td>
<td>Returns an angle in radians whose sine is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Atan</td>
<td>Returns an angle in radians whose tangent is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Atan2</td>
<td>Returns an angle in radians 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 radians</td>
</tr>
<tr>
<td>Sin</td>
<td>Returns the sine of the supplied angle in radians</td>
</tr>
<tr>
<td>Tan</td>
<td>Returns the tangent of the supplied angle in radians</td>
</tr>
<tr>
<td>Cosh</td>
<td>Returns the hyperbolic cosine of the supplied angle in radians</td>
</tr>
<tr>
<td>Sinh</td>
<td>Returns the hyperbolic sine of the supplied angle in radians</td>
</tr>
<tr>
<td>Tanh</td>
<td>Returns the hyperbolic tangent of the supplied angle in radians</td>
</tr>
</tbody>
</table>

OracleDecimal Static (Comparison) Operators

The OracleDecimal static (comparison) operators are listed in Table 14–33.

Table 14–33 OracleDecimal Static (Comparison) Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds two OracleDecimal values</td>
</tr>
<tr>
<td>operator /</td>
<td>Divides one OracleDecimal value by another</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if the two OracleDecimal values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleDecimal values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleDecimal values is greater than or equal to the second</td>
</tr>
</tbody>
</table>
OracleDecimal Members

Table 14–33  (Cont.) OracleDecimal Static (Comparison) Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator !=</td>
<td>Determines if the two OracleDecimal values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleDecimal values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleDecimal values is less than or equal to the second</td>
</tr>
<tr>
<td>operator *</td>
<td>Multiplies two OracleDecimal structures</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts one OracleDecimal structure from another</td>
</tr>
<tr>
<td>operator -</td>
<td>Negates an OracleDecimal structure</td>
</tr>
<tr>
<td>operator%</td>
<td>Returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures.</td>
</tr>
</tbody>
</table>

OracleDecimal Static Operators (Conversion from .NET Type to OracleDecimal)

The OracleDecimal static operators (Conversion from .NET Type to OracleDecimal) are listed in Table 14–34.

Table 14–34  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 14–35.

Table 14–35  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 14–36.
### 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 or not the current instance is an integer</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the current instance has a null value</td>
</tr>
<tr>
<td>IsPositive</td>
<td>Indicates whether or not the current instance is greater than 0</td>
</tr>
<tr>
<td>IsZero</td>
<td>Indicates whether or not 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 14–37.

<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 or not an object is an instance of OracleDecimal, and whether or not 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 System.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 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

OracleDecimal(byte []

This constructor creates a new instance of the OracleDecimal structure and sets its value to the supplied byte array, which is in an Oracle NUMBER format.

Declaration

// C#
public OracleDecimal(byte [] bytes);
Parameters

- **bytes**
  
  A byte array that represents an Oracle NUMBER in an internal Oracle format.

Exceptions

**ArgumentException** - The **bytes** parameter is not in an 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 1-9
- 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.
**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 1-9
- 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**

```csharp
// C#
public OracleDecimal(int intX);
```

**Parameters**

- `intX`
  The provided Int32 value.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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**

```csharp
// 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

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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

```csharp
// C#
public OracleDecimal(long longX);
```

Parameters

- `longX`
  The provided Int64 value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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

```csharp
// 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 1-9
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class” on page 10-2
- "Globalization Support” on page 3-133

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 1-9
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
OracleDecimal Static Fields

The OracleDecimal static fields are listed in Table 14–38.

<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 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>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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;
```
OracleDecimal Structure

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

MaxValue

This static field indicates a constant representing the maximum value for this structure, which is 9.9...9 \times 10^{125} (38 nines followed by 88 zeroes).

Declaration

// C#
public static readonly OracleDecimal MaxValue;

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

MinScale

This static field indicates a constant representing the maximum scale, which is -84.

Declaration

// C#
public static readonly int MinScale;

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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

// C#
public static readonly OracleDecimal MinValue;

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Comparison) Methods

The OracleDecimal static (comparison) methods are listed in Table 14–39.

<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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Equals

This method determines if two OracleDecimal values are equal.

Declaration

// C#
public static bool Equals(OracleDecimal value1, OracleDecimal value2);

Parameters
- value1
  - The first OracleDecimal.
- value2
  - The 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**
  The first OracleDecimal.

- **value2**
  The 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 1-9
- 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**
  The first OracleDecimal.

- **value2**

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure
The 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

**LessThan**
This method determines if the first of two OracleDecimal values is less than the second.

**Declaration**
```csharp
// C#
public static bool LessThan(OracleDecimal value1, OracleDecimal value2);
```

**Parameters**
- `value1`  
The first OracleDecimal.
- `value2`  
The 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 1-9
- 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**
  - The first `OracleDecimal`.
- **value2**
  - The 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 1-9
- `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**
  - The first `OracleDecimal`.
- **value2**
  - The 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 1-9
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Manipulation) Methods

The OracleDecimal static (manipulation) methods are listed in Table 14–40.

<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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Add

This method adds two OracleDecimal structures.

Declaration

// C#  
public static OracleDecimal Add(OracleDecimal val1, OracleDecimal val2);

Parameters

- val1
  
  The first OracleDecimal.
- val2
  
  The 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 1-9
- 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 or not 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 1-9
- 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 1-9
- 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
// C#
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 1-9
- `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 1-9
- `OracleDecimal Members`
- `OracleDecimal Structure`

**Floor**

This method returns a new `OracleDecimal` structure with its value set to the floor of the supplied `OracleDecimal` structure.

**Declaration**

// C#
public static OracleDecimal Floor(OracleDecimal val);

**Parameters**
- `val`
  
  An `OracleDecimal` structure.

**Return Value**
A new `OracleDecimal` structure.

**Remarks**
If either argument has a null value, the returned `OracleDecimal` has a null value.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleDecimal Members`
- `OracleDecimal Structure`

**Max**

This method returns the maximum value of the two supplied `OracleDecimal` structures.

**Declaration**

// C#
public static OracleDecimal Max(OracleDecimal val1, OracleDecimal val2);

**Parameters**
- `val1`
  
  An `OracleDecimal` structure.
- `val2`
An OracleDecimal structure.

**Return Value**

An OracleDecimal structure that has the greater value.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Multiply
This method returns a new OracleDecimal structure with its value set to the result of multiplying two OracleDecimal structures.

Declaration
// C#
public static OracleDecimal Multiply(OracleDecimal val1, OracleDecimal val2);

Parameters
- val1
  An OracleDecimal structure.
- val2
  An OracleDecimal structure.

Return Value
A new OracleDecimal structure.

Remarks
If either argument has a null value, the returned OracleDecimal has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

Negate
This method returns a new OracleDecimal structure with its value set to the negation of the supplied OracleDecimal structures.

Declaration
// C#
public static OracleDecimal Negate(OracleDecimal val);

Parameters
- val
  An OracleDecimal structure.
**Return Value**

A new OracleDecimal structure.

**Remarks**

If either argument has a null value, the returned OracleDecimal has a null value.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

**Parse**

This method converts a string to an OracleDecimal.

**Declaration**

```c#
// 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 1-9
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

**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#
// C#
public static OracleDecimal Round(OracleDecimal val, int decplace);
```
OracleDecimal Static (Manipulation) Methods

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 1-9
- 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 1-9
- 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.
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

OracleNullException - The argument has a null value.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ 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 1-9
■ 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 1-9
- 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 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Logarithmic) Methods

The OracleDecimal static (logarithmic) methods are listed in Table 14–41.

<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 1-9
- 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 1-9
- 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 1-9
- `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 1-9
- `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 1-9
- 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 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Pow(OracleDecimal, int)

This method returns a new OracleDecimal structure with its value set to the supplied OracleDecimal value raised to the supplied Int32 power.

Declaration

// C#
public static OracleDecimal Pow(OracleDecimal val, int power);

Parameters

- val
  An OracleDecimal structure.
- power
  An int value that specifies the power.
Return Value
An OracleDecimal structure.

Remarks
If the supplied OracleDecimal structure has a null value, the returned OracleDecimal has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Trigonometric) Methods

The OracleDecimal static (trigonometric) methods are listed in Table 14–42.

### Table 14–42 OracleDecimal Static (Trigonometric) Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acos</td>
<td>Returns an angle in radians whose cosine is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Asin</td>
<td>Returns an angle in radians whose sine is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Atan</td>
<td>Returns an angle in radians whose tangent is the supplied OracleDecimal structure</td>
</tr>
<tr>
<td>Atan2</td>
<td>Returns an angle in radians 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 radians</td>
</tr>
<tr>
<td>Sin</td>
<td>Returns the sine of the supplied angle in radians</td>
</tr>
<tr>
<td>Tan</td>
<td>Returns the tangent of the supplied angle in radians</td>
</tr>
<tr>
<td>Cosh</td>
<td>Returns the hyperbolic cosine of the supplied angle in radians</td>
</tr>
<tr>
<td>Sinh</td>
<td>Returns the hyperbolic sine of the supplied angle in radians</td>
</tr>
<tr>
<td>Tanh</td>
<td>Returns the hyperbolic tangent of the supplied angle in radians</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

**Acos**

This method returns an angle in radians whose cosine is the supplied OracleDecimal structure.

**Declaration**

```csharp
// C#
public static OracleDecimal Acos(OracleDecimal val);
```

**Parameters**

- `val`  
  An OracleDecimal structure. Range is (-1 to 1).

**Return Value**

An OracleDecimal structure that represents an angle in radians.

**Remarks**

If either argument has a null value, the returned OracleDecimal has a null value.
Asin

This method returns an angle in radians 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 radians.

Remarks

If either argument has a null value, the returned OracleDecimal has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

Atan

This method returns an angle in radians 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 radians.

Remarks

If the argument has a null value, the returned OracleDecimal has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure
Atan2

This method returns an angle in radians whose tangent is the quotient of the two supplied OracleDecimal structures.

**Declaration**

```csharp
// 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 radians.

**Remarks**

If either argument has a null value, the returned OracleDecimal has a null value.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

Cos

This method returns the cosine of the supplied angle in radians.

**Declaration**

```csharp
// C#
public static OracleDecimal Cos(OracleDecimal val);
```

**Parameters**

- `val`:
  An OracleDecimal structure that represents an angle in radians.

**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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Sin

This method returns the sine of the supplied angle in radians.

Declaration

// C#
public static OracleDecimal Sin(OracleDecimal val);

Parameters
- val
  An OracleDecimal structure.

Return Value
An OracleDecimal structure that represents an angle in radians.

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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Tan

This method returns the tangent of the supplied angle in radians.

Declaration

// C#
public static OracleDecimal Tan(OracleDecimal val);

Parameters
- val
  An OracleDecimal structure that represents an angle in radians.

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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Cosh

This method returns the hyperbolic cosine of the supplied angle in radians.

Declaration

// C#
public static OracleDecimal Cosh(OracleDecimal val);

Parameters

- val
  
  An OracleDecimal structure that represents an angle in radians.

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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Sinh

This method returns the hyperbolic sine of the supplied angle in radians.

Declaration

// C#
public static OracleDecimal Sinh(OracleDecimal val);

Parameters

- val
  
  An OracleDecimal structure that represents an angle in radians.

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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Tanh

This method returns the hyperbolic tangent of the supplied angle in radians.

Declaration

```csharp
// C#
public static OracleDecimal Tanh(OracleDecimal val);
```

Parameters

- `val`
  An OracleDecimal structure that represents an angle in radians.

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 1-9
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Static (Comparison) Operators

The OracleDecimal static (comparison) operators are listed in Table 14–43.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator +</td>
<td>Adds two OracleDecimal values</td>
</tr>
<tr>
<td>operator /</td>
<td>Divides one OracleDecimal value by another</td>
</tr>
<tr>
<td>operator ==</td>
<td>Determines if the two OracleDecimal values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines if the first of two OracleDecimal values is greater than the second</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines if the first of two OracleDecimal values is greater than or equal to the second</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines if the two OracleDecimal values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleDecimal values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleDecimal values is less than or equal to the second</td>
</tr>
<tr>
<td>operator *</td>
<td>Multiplies two OracleDecimal structures</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts one OracleDecimal structure from another</td>
</tr>
<tr>
<td>operator -</td>
<td>Negates an OracleDecimal structure</td>
</tr>
<tr>
<td>operator%</td>
<td>Returns a new OracleDecimal structure with its value set to the modulus of two OracleDecimal structures.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

operator +

This method adds two OracleDecimal values.

Declaration

```csharp
// C#
public static OracleDecimal operator + (OracleDecimal val1, OracleDecimal val2);
```

Parameters

- `val1`
  The first OracleDecimal.
- `val2`
  The 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 1-9
- 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
  The first OracleDecimal.
- val2
  The 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 1-9
- 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
  The first OracleDecimal.
- val2
  The second OracleDecimal.
Return Value
Returns true if their values are equal; otherwise, returns false.

Remarks
The following rules apply to the behavior of this method.

- Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
- Two OracleDecimals that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

operator >
This method determines if the first of two OracleDecimal values is greater than the second.

Declaration
// C#
public static bool operator > (OracleDecimal val1, OracleDecimal val2);

Parameters
- val1
  The first OracleDecimal.
- val2
  The second OracleDecimal.

Return Value
Returns true if the two OracleDecimal values are not equal; otherwise, returns false.

Remarks
The following rules apply to the behavior of this method.

- Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
- Two OracleDecimals that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
    The first OracleDecimal.

■ val2
    The 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 1-9

■ 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
    The first OracleDecimal.

■ val2
    The second OracleDecimal.

Return Value

Returns true if the two OracleDecimal values are not equal; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

■ Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
Two OracleDecimals that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

operator <

This method determines if the first of two OracleDecimal values is less than the second.

Declaration

// C#
public static bool operator < (OracleDecimal val1, OracleDecimal val2);

Parameters

- val1
  The first OracleDecimal.
- val2
  The 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

operator <=

This method determines if the first of two OracleDecimal values is less than or equal to the second.

Declaration

// C#
public static bool operator <= (OracleDecimal val1, OracleDecimal val2);

Parameters

- val1
  The first OracleDecimal.
- `val2`

  The 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 1-9
- `OracleDecimal Members`
- `OracleDecimal Structure`

**operator ***

This method multiplies two `OracleDecimal` structures.

**Declaration**

```c#
// C#
public static OracleDecimal operator * (OracleDecimal val1, OracleDecimal val2);
```

**Parameters**

- `val1`

  The first `OracleDecimal`.

- `val2`

  The 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 1-9
- `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**
  The first OracleDecimal.

- **val2**
  The 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 1-9
- 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 1-9
- 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 1-9
- 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 14–44.

Table 14–44  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 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

**implicit operator OracleDecimal(decimal)**

This method returns the OracleDecimal representation of a decimal value.

**Declaration**

```csharp
// 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

**implicit operator OracleDecimal(int)**
This method returns the OracleDecimal representation of an int value.

Declaration

```csharp
// 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

**implicit operator OracleDecimal(long)**
This method returns the OracleDecimal representation of a long value.

Declaration

```csharp
// 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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
OracleDecimal Static Operators (Conversion from OracleDecimal to .NET)

The OracleDecimal static operators (Conversion from OracleDecimal to .NET) are listed in Table 14–45.

Table 14–45  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 1-9
- OracleDecimal Members
- OracleDecimal Structure

**explicit operator byte**

This method returns the byte representation of the OracleDecimal value.

**Declaration**

```csharp
// 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

explicit operator double

This method returns the double representation of the OracleDecimal value.

Declaration

// C#
public static explicit operator double(OracleDecimal val);

Parameters

- **val**
  
  An OracleDecimal structure.

Return Value

A double.

Exceptions

- OracleNullValueException - The OracleDecimal has a null value.
- OverflowException - The double cannot represent the supplied OracleDecimal structure.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

explicit operator short

This method returns the short representation of the OracleDecimal value.

**Declaration**

```
// C#
public static explicit operator short(OracleDecimal val);
```

**Parameters**

- `val`
  
  An OracleDecimal structure.

**Return Value**

A short.

**Exceptions**

- OracleNullValueException - The OracleDecimal has a null value.
- OverFlowException - The short cannot represent the supplied OracleDecimal structure.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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.
explicit operator long

This method returns the long representation of the OracleDecimal value.

Declaration

```csharp
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 1-9
- OracleDecimal Members
- OracleDecimal Structure

explicit operator float

This method returns the float representation of the OracleDecimal value.

Declaration

```csharp
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 1-9
- OracleDecimal Members
- OracleDecimal Structure
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Properties

The OracleDecimal properties are listed in Table 14–46.

Table 14–46  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 or not the current instance is an integer</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the current instance has a null value</td>
</tr>
<tr>
<td>IsPositive</td>
<td>Indicates whether or not the current instance is greater than 0</td>
</tr>
<tr>
<td>IsZero</td>
<td>Indicates whether or not 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 1-9
- 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 1-9
- 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 1-9
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

**IsInt**
This property indicates whether or not 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

**IsNull**
This property indicates whether or not the current instance has a null value.

**Declaration**
```csharp
// C#
```
public bool IsNull {get;}

**Property Value**
A bool value that returns true if the current instance has a null value; otherwise, returns false.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

### IsPositive

This property indicates whether or not the value of the current instance is greater than 0.

**Declaration**

```csharp
// 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

### IsZero

This property indicates whether or not the current instance has a zero value.

**Declaration**

```csharp
// 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.
**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 14-65 for further information.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure
OracleDecimal Instance Methods

The `OracleDecimal` instance methods are listed in Table 14–47.

**Table 14–47 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 or not an object is an instance of <code>OracleDecimal</code>, and whether or not 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 <code>System.Object</code></td>
</tr>
<tr>
<td>ToByte</td>
<td>Returns the <code>byte</code> representation of the current instance</td>
</tr>
<tr>
<td>ToDouble</td>
<td>Returns the <code>double</code> representation of the current instance</td>
</tr>
<tr>
<td>ToInt16</td>
<td>Returns the <code>Int16</code> representation of the current instance</td>
</tr>
<tr>
<td>ToInt32</td>
<td>Returns the <code>Int32</code> representation of the current instance</td>
</tr>
<tr>
<td>ToInt64</td>
<td>Returns the <code>Int64</code> representation of the current instance</td>
</tr>
<tr>
<td>ToSingle</td>
<td>Returns the <code>Single</code> representation of the current instance</td>
</tr>
<tr>
<td>ToString</td>
<td>Overloads <code>Object.ToString()</code> and returns the <code>string</code> representation of the current instance</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

**CompareTo**

This method compares the current instance to the supplied object and returns an integer that represents their relative values.

**Declaration**

// C#
public int CompareTo(object obj);

**Parameters**

- `obj`
  
  The supplied instance.

**Return Value**

The method returns a number:

- Less than zero: if the value of the current instance is less than `obj`.
- `Zero`: if the value of the current instance is equal to `obj`.
- Greater than zero: if the value of the current instance is greater than `obj`. 
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 1-9
- OracleDecimal Members
- OracleDecimal Structure

Equals

Overrides Object

This method determines whether or not an object is an instance of OracleDecimal, and whether or not the value of the object is equal to the current instance.

Declaration

// C#
public override bool Equals(object obj);

Parameters

- obj

An OracleDecimal instance.

Return Value

Returns true if obj is an instance of OracleDecimal, and the value of obj is equal to the current instance; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleDecimal that has a value compares greater than an OracleDecimal that has a null value.
- Two OracleDecimals that contain a null value are equal.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

GetHashCode

Overrides Object

This method returns a hash code for the current instance.

Declaration

// C#
public override int GetHashCode();

Return Value

Returns a hash code.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- 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.
OracleDecimal Structure

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleDecimal Members
- OracleDecimal Structure

**ToInt64**

This method returns the `Int64` representation of the current instance.

**Declaration**

```csharp
// 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

**ToSingle**

This method returns the `Single` representation of the current instance.

**Declaration**

```csharp
// 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 1-9
- OracleDecimal Members
- OracleDecimal Structure

**ToString**

Overriding `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 1-9
- OracleDecimal Members
- OracleDecimal Structure
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
OracleIntervalDS Structure

The OracleIntervalDS structure represents the Oracle INTERVAL DAY TO SECOND data type to be stored in or retrieved from a database. Each OracleIntervalDS stores a period of time in term of days, hours, minutes, seconds, and fractional seconds.

Class Inheritance
System.Object
    System.ValueType
        Oracle.DataAccess.Types.OracleIntervalDS

Declaration
// ADO.NET 2.0: C#
public struct OracleIntervalDS : IComparable, INullable, IXmlSerializable

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() = +000000000 23:59:59.999999999"
        Console.WriteLine("iDSDiff.ToString() = " + iDSDiff.ToString());
    }
}

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 14–48.

Table 14–48  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 14–49.

Table 14–49  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 14–50.

Table 14–50  OracleIntervalDS Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether or not two OracleIntervalDS values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether or not one OracleIntervalDS value is greater than another</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether or not one OracleIntervalDS value is greater than or equal to another</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether or not one OracleIntervalDS value is less than another</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether or not one OracleIntervalDS value is less than or equal to another</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines whether or not 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>
</tbody>
</table>
OracleIntervalDS Members

Table 14–50  OracleIntervalDS Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetPrecision</td>
<td>Returns a new instance of an OracleIntervalDS with the specified day</td>
</tr>
<tr>
<td></td>
<td>precision and fractional second precision</td>
</tr>
</tbody>
</table>

OracleIntervalDS Static Operators

The OracleIntervalDS static operators are listed in Table 14–51.

Table 14–51  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 or not two OracleIntervalDS values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines whether or not one OracleIntervalDS value is greater than another</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines whether or not one OracleIntervalDS value is greater than or</td>
</tr>
<tr>
<td></td>
<td>equal to another</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines whether or not two OracleIntervalDS values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines whether or not one OracleIntervalDS value is less than another</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines whether or not one OracleIntervalDS value is less than or equal</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 14–52.

Table 14–52  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 14–53.
### 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>
<tr>
<td>Hours</td>
<td>Gets the hours component of an OracleIntervalDS</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the current instance has a null value</td>
</tr>
<tr>
<td>Milliseconds</td>
<td>Gets the milliseconds component of an OracleIntervalDS</td>
</tr>
<tr>
<td>Minutes</td>
<td>Gets the minutes component of an OracleIntervalDS</td>
</tr>
<tr>
<td>Nanoseconds</td>
<td>Gets the nanoseconds component of an OracleIntervalDS</td>
</tr>
<tr>
<td>Seconds</td>
<td>Gets the seconds component of an OracleIntervalDS</td>
</tr>
<tr>
<td>TotalDays</td>
<td>Returns the total number, in days, that represent the time period in the OracleIntervalDS structure</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the time interval that is stored in the OracleIntervalDS structure</td>
</tr>
</tbody>
</table>

### OracleIntervalDS Methods

The OracleIntervalDS methods are listed in Table 14–54.

<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 or not 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 System.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 1-9
- 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 1-9
- 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**

```csharp
// C#
public OracleIntervalDS(string intervalStr);
```

**Parameters**

- `intervalStr`  
  A string representing the Oracle `INTERVAL DAY TO SECOND`.

**Exceptions**

- `ArgumentException` - The `intervalStr` parameter is not in the valid format or has an invalid value.
- `ArgumentNullException` - The `intervalStr` parameter is null.

**Remarks**

The value specified in the supplied `intervalStr` must be in Day HH:MI:SSxFF format.

**Example**

"1 2:3:4.99" means 1 day, 2 hours, 3 minutes, 4 seconds, and 990 milliseconds or 1 day, 2 hours, 3 minutes, 4 seconds, and 990000000 nanoseconds.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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**

```csharp
// 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.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members
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 1-9
- 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS(int, int, int, int, int)

This constructor creates a new instance of the OracleIntervalDS structure and sets its value using the supplied days, hours, minutes, seconds, and nanoseconds.

Declaration

// C#
public OracleIntervalDS (int days, int hours, int minutes, int seconds, int nanoseconds);

Parameters

- **days**
  The days provided. Range of day is (-999,999,999 to 999,999,999).

- **hours**
  The hours provided. Range of hour is (-23 to 23).

- **minutes**
  The minutes provided. Range of minute is (-59 to 59).

- **seconds**
  The seconds provided. Range of second is (-59 to 59).

- **nanoseconds**
  The nanoseconds provided. Range of nanosecond is (-999,999,999 to 999,999,999)

Exceptions

- ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.

- ArgumentException - The argument values of the parameters cannot be used to construct a valid OracleIntervalDS.

Remarks

The sign of all the arguments must be the same.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS Static Fields

The OracleIntervalDS static fields are listed in Table 14–55.

<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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

MaxValue

This static field represents the maximum value for an OracleIntervalDS structure.

Declaration

```csharp
// 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

MinValue

This static field represents the minimum value for an OracleIntervalDS structure.

Declaration

```csharp
// C#
public static readonly OracleIntervalDS MinValue;
```
Remarks
Minimum values:
- Day: -999999999
- hour: -23
- minute: -59
- second: -59
- nanosecond: -999999999

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS Static Methods

The OracleIntervalDS static methods are listed in Table 14–56.

Table 14–56  OracleIntervalDS Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether or not two OracleIntervalDS values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether or not one OracleIntervalDS value is greater than another</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether or not one OracleIntervalDS value is greater than or equal to another</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether or not one OracleIntervalDS value is less than another</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether or not one OracleIntervalDS value is less than or equal to another</td>
</tr>
<tr>
<td>NotEquals</td>
<td>Determines whether or not 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

Equals

This static method determines whether or not two OracleIntervalDS values are equal.

Declaration

```csharp
// C#
public static bool Equals(OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- val1
  The first OracleIntervalDS.
- val2
  The 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

GreaterThan
This static method determines whether or not the first of two OracleIntervalDS values is greater than the second.

Declaration
// C#
public static bool GreaterThan(OracleIntervalDS val1, OracleIntervalDS val2);

Parameters
- val1
  The first OracleIntervalDS.
- val2
  The 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

GreaterThanOrEqual
This static method determines whether or not the first of two OracleIntervalDS values is greater than or equal to the second.
Declaration

// C#
public static bool GreaterThanOrEqual(OracleIntervalDS val1,
        OracleIntervalDS val2);

Parameters

■ val1
   The first OracleIntervalDS.

■ val2
   The 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 OracleIntervalDSs that contain a null value are equal.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9

■ OracleIntervalDS Structure

■ OracleIntervalDS Members

LessThan

This static method determines whether or not the first of two OracleIntervalDS
values is less than the second.

Declaration

// C#
public static bool LessThan(OracleIntervalDS val1, OracleIntervalDS val2);

Parameters

■ val1
   The first OracleIntervalDS.

■ val2
   The 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.
OracleIntervalDS Static Methods

- 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

### LessThanOrEqual

This static method determines whether or not the first of two OracleIntervalDS values is less than or equal to the second.

**Declaration**

```csharp
public static bool LessThanOrEqual(OracleIntervalDS val1, OracleIntervalDS val2);
```

**Parameters**

- `val1`
  The first OracleIntervalDS.
- `val2`
  The 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

### NotEquals

This static method determines whether or not two OracleIntervalDS values are not equal.

**Declaration**

```csharp
public static bool NotEquals(OracleIntervalDS val1, OracleIntervalDS val2);
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members
Parameters

■ *val1*
  The first *OracleIntervalDS*.

■ *val2*
  The 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* that contain a null value are equal.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9

■ *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: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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

SetPrecision
This static method returns a new instance of an OracleIntervalDS with the specified day precision and fractional second precision.

Declaration
// C#
public static OracleIntervalDS SetPrecision(OracleIntervalDS value1, int dayPrecision, int fracSecPrecision);

Parameters
- value1
  An OracleIntervalDS structure.
- dayPrecision
  The day precision provided. Range of day precision is (0 to 9).
- fracSecPrecision
  The fractional second precision provided. Range of fractional second precision is (0 to 9).

Return Value
An OracleIntervalDS instance.

Exceptions
ArgumentOutOfRangeException - An argument value is out of the specified range.

Remarks
Depending on the value specified in the supplied dayPrecision, 0 or more leading zeros are displayed in the string returned by ToString().

The value specified in the supplied fracSecPrecision is used to perform a rounding off operation on the supplied OracleIntervalDS value. Depending on this value, 0 or more trailing zeros are displayed in the string returned by ToString().

Example
The OracleIntervalDS with a value of "1 2:3:4.99" results in the string "001 2:3:4.99000" when SetPrecision() is called, with the day precision set to 3 and fractional second precision set to 5.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS Static Operators

The OracleIntervalDS static operators are listed in Table 14–57.

Table 14–57  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 or not two OracleIntervalDS values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines whether or not one OracleIntervalDS value is greater than another</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines whether or not one OracleIntervalDS value is greater than or equal to another</td>
</tr>
<tr>
<td>operator !=</td>
<td>Determines whether or not two OracleIntervalDS values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines whether or not one OracleIntervalDS value is less than another</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines whether or not 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator +

This static operator adds two OracleIntervalDS values.

Declaration

```csharp
// C#
public static OracleIntervalDS operator + (OracleIntervalDS val1,
    OracleIntervalDS val2);
```

Parameters

- val1
  The first OracleIntervalDS.
- val2
  The second OracleIntervalDS.

Return Value
An OracleIntervalDS.
Remarks
If either argument has a null value, the returned OracleIntervalDS structure has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
  The first OracleIntervalDS.
- val2
  The 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 1-9
- 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
  The first OracleIntervalDS.
- val2
  The 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 OracleIntervalDSs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
  The first OracleIntervalDS.
- val2
  The 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 OracleIntervalDSs that contain a null value are equal.
operator !=

This static operator determines if the two OracleIntervalDS values are not equal.

Declaration

```
// C#
public static bool operator != (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- **val1**
  - The first OracleIntervalDS.
- **val2**
  - The 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator <

This static operator determines if the first of two OracleIntervalDS values is less than the second.

Declaration

```
// C#
public static bool operator < (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- **val1**
  - The first OracleIntervalDS.
val2

The 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 OracleIntervalDSs that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
  
The first OracleIntervalDS.
- val2
  
The 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.
operator -

This static operator subtracts one OracleIntervalDS structure from another.

Declaration

```
// C#
public static OracleIntervalDS operator - (OracleIntervalDS val1, OracleIntervalDS val2);
```

Parameters

- `val1`  
The first OracleIntervalDS.
- `val2`  
The 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

operator -

This static operator negates the supplied OracleIntervalDS structure.

Declaration

```
// C#
public static OracleIntervalDS operator - (OracleIntervalDS val);
```

Parameters

- `val`  
An OracleIntervalDS.

Return Value

An OracleIntervalDS structure.
**Remarks**

If the supplied `OracleIntervalDS` structure has a null value, the returned `OracleIntervalDS` structure has a null value.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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`  
  The 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 1-9  
- `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`  
  The 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 1-9
- `OracleIntervalDS` Structure
- `OracleIntervalDS Members`
OracleIntervalDS Type Conversions

The *OracleIntervalDS* type conversions are listed in Table 14–58.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator TimeSpan</td>
<td>Converts an <em>OracleIntervalDS</em> structure to a <em>TimeSpan</em> structure</td>
</tr>
<tr>
<td>explicit operator OracleIntervalDS</td>
<td>Converts a string to an <em>OracleIntervalDS</em> structure</td>
</tr>
<tr>
<td>implicit operator OracleIntervalDS</td>
<td>Converts a <em>TimeSpan</em> structure to an <em>OracleIntervalDS</em> structure</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- *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 1-9
- *OracleIntervalDS* Structure
- *OracleIntervalDS* Members

### explicit operator OracleIntervalDS

This type conversion operator converts a string to an *OracleIntervalDS* structure.
Declaration

// C#
public static explicit operator OracleIntervalDS (string intervalStr);

Parameters

- `intervalStr`
  A string representation of an Oracle INTERVAL DAY TO SECOND.

Return Value

An OracleIntervalDS structure.

Exceptions

- ArgumentException - The supplied `intervalStr` parameter is not in the correct format or has an invalid value.
- ArgumentNullException - The `intervalStr` parameter is null.

Remarks

The returned OracleIntervalDS structure contains the same time interval represented by the supplied `intervalStr`. The value specified in the supplied `intervalStr` must be in Day HH:MI:SSxFF format.

Example

"1 2:3:4.99" means 1 day, 2 hours, 3 minutes 4 seconds and 990 milliseconds or 1 day, 2 hours, 3 minutes 4 seconds and 990000000 nanoseconds.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members
OracleIntervalDS Properties

The OracleIntervalDS properties are listed in Table 14–59.

### Table 14–59  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 INTERVAL DAY TO SECOND in Oracle internal format</td>
</tr>
<tr>
<td>Days</td>
<td>Gets the days component of an OracleIntervalDS</td>
</tr>
<tr>
<td>Hours</td>
<td>Gets the hours component of an OracleIntervalDS</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the current instance has a null value</td>
</tr>
<tr>
<td>Milliseconds</td>
<td>Gets the milliseconds component of an OracleIntervalDS</td>
</tr>
<tr>
<td>Minutes</td>
<td>Gets the minutes component of an OracleIntervalDS</td>
</tr>
<tr>
<td>Nanoseconds</td>
<td>Gets the nanoseconds component of an OracleIntervalDS</td>
</tr>
<tr>
<td>Seconds</td>
<td>Gets the seconds component of an OracleIntervalDS</td>
</tr>
<tr>
<td>TotalDays</td>
<td>Returns the total number, in days, that represent the time period in the OracleIntervalDS structure</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the time interval that is stored in the OracleIntervalDS structure</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

### BinData

This property returns an array of bytes that represents the Oracle INTERVAL DAY TO SECOND in Oracle internal format.

**Declaration**

```
// C#
public byte[] BinData {get;}
```

**Property Value**

A byte array that represents an Oracle INTERVAL DAY TO SECOND in Oracle internal format.

**Exceptions**

OracleNullValueException - The current instance has a null value.
Remarks

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

IsNull

This property indicates whether or not 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

Milliseconds
This property gets the milliseconds component of an OracleIntervalDS.

Declaration
// C#
public double Milliseconds {get;}

Property Value
A double that represents milliseconds component.

Exceptions
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

Minutes
This property gets the minutes component of an OracleIntervalDS.

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

Property Value
A int that represents minutes component.

Exceptions
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

Nanoseconds
This property gets the nanoseconds component of an OracleIntervalDS.
OracleIntervalDS Properties

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 1-9
- 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 1-9
- 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.
Value

This property specifies the time interval that is stored in the **OracleIntervalDS** structure.

**Declaration**

```csharp
// 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 1-9
- **OracleIntervalDS Structure**
- **OracleIntervalDS Members**
OracleIntervalDS Methods

The OracleIntervalDS methods are listed in Table 14–60.

<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 or not 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 System.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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

CompareTo

This method compares the current OracleIntervalDS instance to an object, and returns an integer that represents their relative values.

Declaration

// C#
public int CompareTo(object obj);

Parameters

- **obj**
  The object being compared to.

Return Value

The method returns:

- Less than zero: if the current OracleIntervalDS represents a shorter time interval than **obj**.
- Zero: if the current OracleIntervalDS and **obj** represent the same time interval.
- Greater than zero: if the current OracleIntervalDS represents a longer time interval than **obj**.

Implements

IComparable

Exceptions

ArgumentException - The **obj** parameter is not of type OracleIntervalDS.
Remarks
The following rules apply to the behavior of this method.

- The comparison must be between 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 1-9
- OracleIntervalDS Structure
- OracleIntervalDS Members

Equals
This method determines whether or not 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 1-9
- 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 1-9
- `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 1-9
- `OracleIntervalDS` Structure
- `OracleIntervalDS` Members
**OracleIntervalYM Structure**

The `OracleIntervalYM` structure represents the Oracle `INTERVAL YEAR TO MONTH` data type to be stored in or retrieved from a database. Each `OracleIntervalYM` stores a period of time in years and months.

**Class Inheritance**

`System.Object`  
`System.ValueType`  
`Oracle.DataAccess.Types.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() = +00000001-00"  
        Console.WriteLine("iYMDiff.ToString() = " + iYMDiff.ToString());  
    }  
}

**Requirements**

Namespace: `Oracle.DataAccess.Types`  
Assembly: `Oracle.DataAccess.dll`  
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 14–61.

Table 14–61 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 14–62.

Table 14–62 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 14–63.

Table 14–63 OracleIntervalYM Static Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether or not two OracleIntervalYM values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether or not one OracleIntervalYM value is greater than another</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether or not one OracleIntervalYM value is greater than or equal to another</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether or not one OracleIntervalYM value is less than another</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether or not 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>
</tbody>
</table>
### OracleIntervalYM Members

**Table 14–63  (Cont.) OracleIntervalYM Static Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 14–64.

**Table 14–64  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 or not two OracleIntervalYM values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines whether or not one OracleIntervalYM value is greater than another</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines whether or not 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 or not one OracleIntervalYM value is less than another</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines whether or not 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 14–65.

**Table 14–65  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 14–66.
**Table 14–66  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 or not 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>

**OracleIntervalYM Methods**

The OracleIntervalYM methods are listed in Table 14–67.

**Table 14–67  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 or not 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 System.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 1-9
- 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 1-9
- 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**

```csharp
// 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**

IllegalArgumentException - The totalMonths parameter is out of the specified range.
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.

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 \(-1,000,000,000 < totalYears > 1,000,000,000\).
OracleIntervalYM Constructors

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 1-9
- 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 1-9
- 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);
OracleIntervalYM Structure

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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Static Fields

The OracleIntervalYM static fields are listed in Table 14–68.

<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 1-9
- 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 1-9
- 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.
Null

This static field represents a null value that can be assigned to an OracleIntervalYM instance.

Declaration

```csharp
// C#
public static readonly OracleIntervalYM Null;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

Zero

This static field represents a zero value for an OracleIntervalYM structure.

Declaration

```csharp
// C#
public static readonly OracleIntervalDS Zero;
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Static Methods

The OracleIntervalYM static methods are listed in Table 14–69.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether or not two OracleIntervalYM values are equal (Overloaded)</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Determines whether or not one OracleIntervalYM value is greater than another</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether or not one OracleIntervalYM value is greater than or equal to another</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether or not one OracleIntervalYM value is less than another</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether or not 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

Equals

This static method determines whether or not 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.

- 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

GreaterThan
This static method determines whether or not the first of two OracleIntervalYM values is greater than the second.

Declaration
// C# public static bool GreaterThan(OracleIntervalYM val1, OracleIntervalYM val2);

Parameters
- val1
  The first OracleIntervalYM.
- val2
  The 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

GreaterThanOrEqual
This static method determines whether or not the first of two OracleIntervalYM values is greater than or equal to the second.

Declaration
// C#
public static bool GreaterThanOrEqual(OracleIntervalYM val1,
OracleIntervalYM val2);

**Parameters**

- **val1**
  The first `OracleIntervalYM`.
- **val2**
  The 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 1-9
- `OracleIntervalYM` Structure
- `OracleIntervalYM` Members

---

This static method determines whether or not the first of two `OracleIntervalYM` values is less than the second.

**Declaration**

```
// C#
public static bool LessThan(OracleIntervalYM val1, OracleIntervalYM val2);
```

**Parameters**

- **val1**
  The first `OracleIntervalYM`.
- **val2**
  The 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 structures that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

LessThanOrEqual

This static method determines whether or not 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`
  The first OracleIntervalYM.
- `val2`
  The 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 structures that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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`
  The first OracleIntervalYM.
The 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 `OracleIntervalYM`s that contain a null value are equal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleIntervalYM` Structure
- `OracleIntervalYM` Members

### Parse

This static method returns an `OracleIntervalYM` structure and sets its value for time interval using a string.

**Declaration**

```csharp
// 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.
SetPrecision

This static method returns a new instance of an OracleIntervalYM with the specified year precision.

Declaration

```csharp
// 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.
OracleIntervalYM Static Operators

The OracleIntervalYM static operators are listed in Table 14–70.

<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 or not two OracleIntervalYM values are equal</td>
</tr>
<tr>
<td>operator &gt;</td>
<td>Determines whether or not one OracleIntervalYM value is greater than another</td>
</tr>
<tr>
<td>operator &gt;=</td>
<td>Determines whether or not 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 or not one OracleIntervalYM value is less than another</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines whether or not 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 1-9
- 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**
  - The first OracleIntervalYM.
- **val2**
  - The 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator ==
This static operator determines if two OracleIntervalYM values are equal.

Declaration
// C#
public static bool operator == (OracleIntervalYM val1, OracleIntervalYM val2);

Parameters
- val1
  The first OracleIntervalYM.
- val2
  The 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 OracleIntervalYMs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

operator >
This static operator determines if the first of two OracleIntervalYM values is greater than the second.

Declaration
// C#
public static bool operator > (OracleIntervalYM val1, OracleIntervalYM val2);

Parameters
- val1
  The first OracleIntervalYM.
OracleIntervalYM Static Operators

- **val2**
  The 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 1-9
- 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**

```csharp
// C#
public static bool operator >= (OracleIntervalYM val1, OracleIntervalYM val2);
```

**Parameters**

- **val1**
  The first OracleIntervalYM.
- **val2**
  The 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members
operator !=

This static operator determines whether two OracleIntervalYM values are not equal.

Declaration

// C#
public static bool operator != (OracleIntervalYM val1, OracleIntervalYM val2)

Parameters

■ val1
  The first OracleIntervalYM.
■ val2
  The 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 OracleIntervalYMs that contain a null value are equal.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ OracleIntervalYM Structure
■ OracleIntervalYM Members

operator <

This static operator determines if the first of two OracleIntervalYM values is less than the second.

Declaration

// C#
public static bool operator < (OracleIntervalYM val1, OracleIntervalYM val2);

Parameters

■ val1
  The first OracleIntervalYM.
■ val2
  The 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 1-9
- 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
  The first OracleIntervalYM.
- val2
  The 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 1-9
- 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**
  The first OracleIntervalYM.
- **val2**
  The 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 1-9
- 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

**operator ***

This static operator multiplies an OracleIntervalYM value by a number.
Declaration

```csharp
// C#
public static OracleIntervalYM operator * (OracleIntervalYM val1, int multiplier);
```

Parameters

- `val1`
The 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 1-9
- `OracleIntervalYM Structure`
- `OracleIntervalYM Members`

operator /

This static operator divides an `OracleIntervalYM` value by a number.

Declaration

```csharp
// C#
public static OracleIntervalYM operator / (OracleIntervalYM val1, int divisor);
```

Parameters

- `val1`
The 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 1-9
- `OracleIntervalYM Structure`
- `OracleIntervalYM Members`
OracleIntervalYM Type Conversions

The OracleIntervalYM conversions are listed in Table 14–71.

Table 14–71  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>implicit operator OracleIntervalYM</td>
<td>Converts the number of months to an OracleIntervalYM structure</td>
</tr>
<tr>
<td>explicit operator OracleIntervalYM</td>
<td>Converts a string to an OracleIntervalYM structure</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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

```csharp
// 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

explicit operator OracleIntervalYM

This type conversion operator converts the string `intervalStr` to an OracleIntervalYM structure.
Declaration

// C#
public static explicit operator OracleIntervalYM (string intervalStr);

Parameters

- intervalStr

  A string representation of an Oracle INTERVAL YEAR TO MONTH.

Return Value

An OracleIntervalYM structure.

Exceptions

- ArgumentException - The supplied intervalStr parameter is not in the correct format or has an invalid value.
- ArgumentNullException - The intervalStr parameter is null.

Remarks

The returned OracleIntervalDS structure contains the same time interval represented by the supplied intervalStr. The value specified in the supplied intervalStr must be in Year-Month format.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Properties

The OracleIntervalYM properties are listed in Table 14–72.

Table 14–72  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 or not 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

BinData

This property returns an array of bytes that represents the Oracle INTERVAL YEAR TO MONTH in an 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

IsNull

This property indicates whether or not 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 1-9
- 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 1-9
- 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members
Value
This property gets the total number of months that is stored in the OracleIntervalYM structure.

Declaration
// C#
public long Value {get;}

Property Value
The total number of months representing the time interval.

Exceptions
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

Years
This property gets the years component of an OracleIntervalYM.

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

Property Value
An int representing the years component.

Exceptions
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members
OracleIntervalYM Methods

The OracleIntervalYM methods are listed in Table 14–73.

<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 or not 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 System.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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

CompareTo

This method compares the current OracleIntervalYM instance to the supplied object, and returns an integer that represents their relative values.

Declaration

```csharp
// 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.
OracleIntervalYM Methods

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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members

Equals

Overrides Object
This method determines whether or not 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 1-9
- 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 1-9
- 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 1-9
- OracleIntervalYM Structure
- OracleIntervalYM Members
The **OracleString** structure represents a variable-length stream of characters to be stored in or retrieved from a database.

**Class Inheritance**

```
System.Object
    System.ValueType
        Oracle.DataAccess.Types.OracleString
```

**Declaration**

```
// ADO.NET 2.0: C#
public struct OracleString : IComparable, INullable, IXmlSerializable
```

**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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 14–74.

Table 14–74  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 14–75.

Table 14–75  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 14–76.

Table 14–76  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 or not the first of two OracleString values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether or not the first of two OracleString values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether or not the first of two OracleString values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether or not 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 14–77.
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 14–78.

<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 14–79.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsCaseIgnored</td>
<td>Indicates whether or not case should be ignored when performing string comparison</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not 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 14–80.

<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>
### Table 14–80  (Cont.) 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 or not 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 System.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 1-9
- 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 1-9
- OracleString Structure
- OracleString Members

OracleString(string)

This constructor creates a new instance of the OracleString structure and sets its value using a string.

Declaration

// C#
public OracleString(string data);

Parameters

- data
A string value.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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 1-9
- `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.
OracleString Structure

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 1-9
- 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
  See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members
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**

```csharp
// 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 1-9
- OracleString Structure
- OracleString Members
OracleString Static Fields

The `OracleString` static fields are listed in Table 14–81.

<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 1-9
- `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
// C#
public static readonly OracleString Null;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleString` Structure
- `OracleString` Members
OracleString Static Methods

The OracleString static methods are listed in Table 14–82.

Table 14–82  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 or not the first of two OracleString values is greater than the second</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Determines whether or not the first of two OracleString values is greater than or equal to the second</td>
</tr>
<tr>
<td>LessThan</td>
<td>Determines whether or not the first of two OracleString values is less than the second</td>
</tr>
<tr>
<td>LessThanOrEqual</td>
<td>Determines whether or not 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 1-9
- 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`
  The first OracleString.
- `str2`
  The 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 or not the two OracleStrings being compared are equal.

Declaration

```csharp
// C#
public static bool Equals(OracleString str1, OracleString str2);
```

Parameters

- `str1`  
  The first OracleString.
- `str2`  
  The second OracleString.

Return Value

Returns `true` if the two OracleStrings being compared are equal; returns `false` otherwise.

Remarks

The following rules apply to the behavior of this method.

- Any OracleString that has a value is greater than an OracleString that has a null value.
- Two OracleStrings that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

GreaterThan

This static method determines whether or not the first of two OracleString values is greater than the second.

Declaration

```csharp
// C#
public static bool GreaterThan(OracleString str1, OracleString str2);
```

Parameters

- `str1`  
  The first OracleString.
str2

The 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 1-9
- OracleString Structure
- OracleString Members

GreaterThanOrEqual
This static method determines whether or not the first of two OracleString values is greater than or equal to the second.

Declaration
// C#
public static bool GreaterThanOrEqual(OracleString str1,
    OracleString str2);

Parameters
- str1
  The first OracleString.
- str2
  The 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.
OracleString Static Methods

LessThan

This static method determines whether or not the first of two OracleString values is less than the second.

Declaration

```c#
public static bool LessThan(OracleString str1, OracleString str2);
```

Parameters

- `str1` The first OracleString.
- `str2` The 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 OracleStrings that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

LessThanOrEqual

This static method determines whether or not the first of two OracleString values is less than or equal to the second.

Declaration

```c#
public static bool LessThanOrEqual(OracleString str1, OracleString str2);
```

Parameters

- `str1` The first OracleString.
- `str2` The 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 1-9
- OracleString Structure
- OracleString Members
The 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 1-9
- 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**
  The first OracleString.
- **str2**
  The 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 1-9
- OracleString Structure
- OracleString Members
**OracleString Static Operators**

The **OracleString** static operators are listed in Table 14–83.

<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 1-9
- OracleString Structure
- OracleString Members

**operator +**

This static operator concatenates two **OracleString** values.

**Declaration**

// C#
public static OracleString operator + (OracleString value1, OracleString value2);

**Parameters**
- value1
  - The first OracleString.
- value2
  - The second OracleString.

**Return Value**

An **OracleString**.

**Remarks**

If either argument has a null value, the returned **OracleString** structure has a null value.
operator ==

This static operator determines if two OracleString values are equal.

Declaration

```csharp
// C#
public static bool operator == (OracleString value1, OracleString value2);
```

Parameters

- `value1`  
The first OracleString.
- `value2`  
The 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 1-9
- OracleString Structure
- OracleString Members

operator >

This static operator determines if the first of two OracleString values is greater than the second.

Declaration

```csharp
// C#
public static bool operator > (OracleString value1, OracleString value2);
```

Parameters

- `value1`  
The first OracleString.
- `value2`  
The 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 1-9
- 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
  The first OracleString.
- value2
  The 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 1-9
- OracleString Structure
- OracleString Members

operator !=

This static operator determines if two OracleString values are not equal.
Declaration

```csharp
// C#
public static bool operator != (OracleString value1, OracleString value2);
```

**Parameters**
- `value1`
  The first `OracleString`.
- `value2`
  The 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 `OracleString`s that contain a null value are equal.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleString Structure`
- `OracleString Members`

```
operator <
```

This static operator determines if the first of two `OracleStrings` is less than the second.

Declaration

```csharp
// C#
public static bool operator < (OracleString value1, OracleString value2);
```

**Parameters**
- `value1`
  The first `OracleString`.
- `value2`
  The 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` that has a null value.
Two OracleStrings that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

**operator <=**

This static operator determines if the first of two OracleString values is less than or equal to the second.

**Declaration**

```
// C#
public static bool operator <= (OracleString value1, OracleString value2);
```

**Parameters**

- **value1**
  - The first OracleString.
- **value2**
  - The 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 1-9
- OracleString Structure
- OracleString Members
### OracleString Type Conversions

The OracleString type conversions are listed in Table 14–84.

**Table 14–84  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 1-9
- OracleString Structure
- OracleString Members

#### explicit operator string

This type conversion operator converts the supplied OracleString to a string.

**Declaration**

```csharp
// 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 1-9
- OracleString Structure
- OracleString Members

#### implicit operator OracleString

This type conversion operator converts the supplied string to an OracleString.

**Declaration**

```csharp
// 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 1-9
- OracleString Structure
- OracleString Members
OracleString Properties

The OracleString properties are listed in Table 14–85.

Table 14–85 OracleString Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsCaseIgnored</td>
<td>Indicates whether or not case should be ignored when performing string comparison</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not 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>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

**IsCaseIgnored**

This property indicates whether or not 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 nonzero value
Console.WriteLine(string1.CompareTo(string2));

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

IsNull

This property indicates whether or not the current instance contains a null value.

Declaration
// C#
public bool IsNull {get;}

Property Value
Returns true if the current instance contains has a null value; otherwise, returns false.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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.
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 1-9
- OracleString Structure
- OracleString Members
The `OracleString` methods are listed in Table 14–86.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Returns a copy of the current <code>OracleString</code> instance</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Compares the current <code>OracleString</code> instance to the supplied object, and returns an integer that represents their relative values</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether or not an object has the same string value as the current <code>OracleString</code> structure (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns a hash code for the <code>OracleString</code> instance</td>
</tr>
<tr>
<td>GetNonUnicodeBytes</td>
<td>Returns an array of bytes, containing the contents of the <code>OracleString</code>, in the client character set format</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from <code>System.Object</code></td>
</tr>
<tr>
<td>GetUnicodeBytes</td>
<td>Returns an array of bytes, containing the contents of the <code>OracleString</code>, in Unicode format</td>
</tr>
<tr>
<td>ToString</td>
<td>Converts the current <code>OracleString</code> instance to a string</td>
</tr>
</tbody>
</table>

### See Also:
- "`Oracle.DataAccess.Types Namespace`" on page 1-9
- `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 1-9
- OracleString Structure
- OracleString Members

**CompareTo**

This method compares the current OracleString instance to the supplied object, and returns an integer that represents their relative values.

**Declaration**

// C#
public int CompareTo(object obj);

**Parameters**

- **obj**
  The object being compared to the current instance.

**Return Value**

The method returns a number that is:
- Less than zero: if the current OracleString value is less than obj.
- Zero: if the current OracleString value is equal to obj.
- Greater than zero: if the current OracleString value is greater than obj.

**Implements**

IComparable

**Exceptions**

ArgumentException - The obj parameter is not of type OracleString.

**Remarks**

The following rules apply to the behavior of this method.
- The comparison must be between 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.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

Equals

This method determines whether or not 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.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

GetHashCode

Overrides Object

This method returns a hash code for the OracleString instance.

Declaration

// C#
public override int GetHashCode();

Return Value

A number that represents the hash code.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

GetNonUnicodeBytes
This method returns an array of bytes, containing the contents of the OracleString, in the client character set format.

Declaration

// C#
public byte[] GetNonUnicodeBytes();

Return Value
A byte array that contains the contents of the OracleString in the client character set format.

Remarks
If the current instance has a null value, an OracleNullValueException is thrown.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleString Structure
- OracleString Members

GetUnicodeBytes
This method returns an array of bytes, containing the contents of the OracleString in Unicode format.

Declaration

// C#
public byte[] GetUnicodeBytes();

Return Value
A byte array that contains the contents of the OracleString in Unicode format.

Remarks
If the current instance has a null value, an OracleNullValueException is thrown.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleString Structure
- OracleString Members
OracleTimeStamp Structure

The OracleTimeStamp structure represents the Oracle TIMESTAMP data type 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
System.Object
    System.ValueType
        Oracle.DataAccess.Types.OracleTimeStamp

Declaration

// ADO.NET 2.0: C#
public struct OracleTimeStamp : IComparable, INullable, IXmlSerializable

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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 14–87

Table 14–87 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 14–88.

Table 14–88 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 14–89.

Table 14–89 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 Members

### OracleTimeStamp Static Operators

The OracleTimeStamp static operators are listed in Table 14–90.

#### Table 14–90 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 OracleDateTime values are not equal</td>
</tr>
<tr>
<td>operator &lt;</td>
<td>Determines if the first of two OracleDateTime values is less than the second</td>
</tr>
<tr>
<td>operator &lt;=</td>
<td>Determines if the first of two OracleDateTime values is less than or equal to the second</td>
</tr>
<tr>
<td>operator -</td>
<td>Subtracts the supplied instance value from the supplied OracleDateTime and returns a new OracleDateTime structure (Overloaded)</td>
</tr>
</tbody>
</table>

### OracleTimeStamp Static Type Conversions

The OracleDateTime static type conversions are listed in Table 14–91.

#### Table 14–91 OracleDateTime Static Type Conversions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit operator OracleDateTime</td>
<td>Converts an instance value to an OracleDateTime structure (Overloaded)</td>
</tr>
<tr>
<td>implicit operator OracleDateTime</td>
<td>Converts an instance value to an OracleDateTime structure (Overloaded)</td>
</tr>
<tr>
<td>explicit operator DateTime</td>
<td>Converts an OracleDateTime value to a DateTime structure</td>
</tr>
</tbody>
</table>

### OracleDateTime Properties

The OracleDateTime properties are listed in Table 14–92.

#### Table 14–92 OracleDateTime 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 OracleTIMESTAMP in Oracle internal format</td>
</tr>
</tbody>
</table>
OracleTimeStamp Structure

**Table 14–93 (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 or not 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 14–93.

**Table 14–93 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 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>
</tbody>
</table>
Table 14–93 (Cont.) OracleTimeStamp Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether or not 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 System.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 1-9
- OracleTimeStamp Structure
OracleTimeStamp Constructors

The OracleTimeStamp constructors create new instances of the OracleTimeStamp structure.

Overload List:
- OracleTimeStamp(DateTime)
  This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using the supplied DateTime value.
- OracleTimeStamp(string)
  This constructor creates a new instance of the OracleTimeStamp structure and sets its value using the supplied string.
- OracleTimeStamp(int, int, int)
  This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date using year, month, and day.
- OracleTimeStamp(int, int, int, int, int, int)
  This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, and second.
- OracleTimeStamp(int, int, int, int, int, int, double)
  This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.
- OracleTimeStamp(int, int, int, int, int, int, int)
  This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.
- OracleTimeStamp(byte [ ])
  This constructor creates a new instance of the OracleTimeStamp structure and sets its value to the provided byte array, which is in the internal Oracle TIMESTAMP format.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 Constructos

Parameters

- **dt**
  
The supplied DateTime value.

Exceptions

ArgumentException - The dt parameter cannot be used to construct a valid OracleTimeStamp.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(string)

This constructor creates a new instance of the OracleTimeStamp structure and sets its value using the supplied string.

Declaration

// C#
public OracleTimeStamp (string tsStr);

Parameters

- **tsStr**
  
  A string that represents an Oracle TIMESTAMP.

Exceptions

ArgumentException - The tsStr value is an invalid string representation of an Oracle TIMESTAMP or the supplied tsStr is not in the timestamp format specified by the OracleGlobalization.TimeStampFormat property of the thread, which represents the Oracle 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
- 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 Constructors

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(int, int, int, int, int, int)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration
// C#
public OracleTimeStamp (int year, int month, int day, int hour,
int minute, int second);

Parameters
■ year
The year provided. Range of year is (-4712 to 9999).
■ month
The month provided. Range of month is (1 to 12).
■ day
The day provided. Range of day is (1 to 31).
■ hour
The hour provided. Range of hour is (0 to 23).
■ minute
The minute provided. Range of minute is (0 to 59).
■ second
The second provided. Range of second is (0 to 59).

Exceptions
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, double)
This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.
Declaration

// C#
public OracleTimeStamp(int year, int month, int day, int hour,
    int minute, int second, double millisecond);

Parameters

- **year**
  The year provided. Range of year is (-4712 to 9999).
- **month**
  The month provided. Range of month is (1 to 12).
- **day**
  The day provided. Range of day is (1 to 31).
- **hour**
  The hour provided. Range of hour is (0 to 23).
- **minute**
  The minute provided. Range of minute is (0 to 59).
- **second**
  The second provided. Range of second is (0 to 59).
- **milliSeconds**
  The milliseconds provided. Range of millisecond is (0 to 999.999999).

Exceptions

- 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(int, int, int, int, int, int, int)

This constructor creates a new instance of the OracleTimeStamp structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

Declaration

// C#
public OracleTimeStamp (int year, int month, int day, int hour,
    int minute, int second, int nanosecond);}
Parameters

- **year**
  The year provided. Range of *year* is (-4712 to 9999).

- **month**
  The month provided. Range of *month* is (1 to 12).

- **day**
  The day provided. Range of *day* is (1 to 31).

- **hour**
  The hour provided. Range of *hour* is (0 to 23).

- **minute**
  The minute provided. Range of *minute* is (0 to 59).

- **second**
  The second provided. Range of *second* is (0 to 59).

- **nanosecond**
  The nanosecond provided. Range of *nanosecond* is (0 to 999999999).

Exceptions

**ArgumentOutOfRangeException** - The argument value for one or more of the parameters is out of the specified range.

**ArgumentException** - The argument values of the parameters cannot be used to construct a valid OracleTimeStamp (that is, the day is out of range for the month).

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

OracleTimeStamp(byte [])

This constructor creates a new instance of the OracleTimeStamp structure and sets its value to the provided byte array, which is in the internal Oracle TIMESTAMP format.

Declaration

```c#
// 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Static Fields

The OracleTimeStamp static fields are listed in Table 14–94.

<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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Static Methods

The OracleTimeStamp static methods are listed in Table 14–95.

<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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

Equals

This static method determines if two OracleTimeStamp values are equal.

Declaration

```csharp
// C#
public static bool Equals(OracleTimeStamp value1, OracleTimeStamp value2);
```

Parameters

- `value1`  
The first OracleTimeStamp.
- `value2`  
The second OracleTimeStamp.

Return Value

Returns `true` if two OracleTimeStamp values are equal; otherwise, returns `false`. 
Remarks
The following rules apply to the behavior of this method.

- Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
- Two `OracleTimeStamp`s that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`

**GreaterThan**
This static method determines if the first of two `OracleTimeStamp` values is greater than the second.

**Declaration**
```csharp
// C#
public static bool GreaterThan(OracleTimeStamp value1, OracleTimeStamp value2);
```

**Parameters**
- `value1`
  The first `OracleTimeStamp`.
- `value2`
  The 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 `OracleTimeStamp`s that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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
    The first OracleTimeStamp.

■ value2
    The 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 1-9
■ 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
    The first OracleTimeStamp.

■ value2
    The second OracleTimeStamp.

Return Value

Returns true if the first of two OracleTimeStamp values is less than the second.
Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.
■ Any `OracleTimeStamp` that has a value is greater than an `OracleTimeStamp` that has a null value.
■ Two `OracleTimeStamps` that contain a null value are equal.

See Also:
■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ `OracleTimeStamps Structure`
■ `OracleTimeStamps Members`

LessThanOrEqual
This static method determines if the first of two `OracleTimeStamps` values is less than or equal to the second.

Declaration

```
// C#
public static bool LessThanOrEqual(OracleTimeStamp value1,
                                 OracleTimeStamp value2);
```

Parameters
■ value1
  The first `OracleTimeStamp`.
■ value2
  The second `OracleTimeStamp`.

Return Value
Returns `true` if the first of two `OracleTimeStamps` values is less than or equal to the second. Returns `false` otherwise.

Remarks
The following rules apply to the behavior of this method.
■ Any `OracleTimeStamps` that has a value is greater than an `OracleTimeStamps` that has a null value.
■ Two `OracleTimeStamps` that contain a null value are equal.

See Also:
■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ `OracleTimeStamps Structure`
■ `OracleTimeStamps Members`

NotEquals
This static method determines if two `OracleTimeStamps` values are not equal.

Declaration

```
// C#
public static bool NotEquals(OracleTimeStamp value1,
                            OracleTimeStamp value2);
```
OracleTimeStamp Static Methods

Parameters

■ value1
   The first OracleTimeStamp.

■ value2
   The second OracleTimeStamp.

Return Value

Returns true if two OracleTimeStamp values are not equal. Returns false otherwise.

Remarks

The following rules apply to the behavior of this method.

■ Any OracleTimeStamp that has a value is greater than an OracleTimeStamp that has a null value.

■ Two OracleTimeStamps that contain a null value are equal.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9

■ OracleTimeStamp Structure

■ OracleTimeStamp Members

GetSysDate

This static method gets an OracleTimeStamp structure that represents the current date and time.

Declaration

// C#
public static OracleTimeStamp GetSysDate();

Return Value

An OracleTimeStamp structure that represents the current date and time.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9

■ OracleTimeStamp Structure

■ OracleTimeStamp Members

Parse

This static method gets an OracleTimeStamp structure and sets its value using the supplied string.

Declaration

// C#
public static OracleTimeStamp Parse(string datetime);

Parameters

■ datetime
A string that represents an Oracle TIMESTAMP.

**Return Value**
An OracleTimeStamp structure.

**Exceptions**
ArgumentException - The tsStr is an invalid string representation of an Oracle TIMESTAMP or the supplied tsStr is not in the timestamp format specified by the OracleGlobalization.TimeStampFormat property of the thread, which represents the Oracle 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
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 1-9
■ OracleTimeStamp Structure
■ OracleTimeStamp Members
OracleTimeStamp Static Operators

The OracleTimeStamp static operators are listed in Table 14–96.

Table 14–96 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 1-9
- 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

// 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator + (OracleTimeStamp, OracleIntervalYM)

This static operator adds the supplied OracleIntervalYM to the supplied OracleTimeStamp and returns a new OracleTimeStamp structure.

Declaration

// C#
public static operator + (OracleTimeStamp value1, OracleIntervalYM value2);

Parameters

- value1
  An OracleTimeStamp.
- value2
  An OracleIntervalYM.

Return Value

An OracleTimeStamp.
Remarks
If either parameter has a null value, the returned OracleTimeStamp has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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
  The first OracleTimeStamp.
- value2
  The 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 1-9
- 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
  The first OracleTimeStamp.
- value2
  The 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 1-9
- 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

```csharp
// C#
public static bool operator >= (OracleTimeStamp value1,
     OracleTimeStamp value2);
```

Parameters

- `value1`:
  The first `OracleTimeStamp`.

- `value2`:
  The 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 `OracleTimeStamp` s that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStamp` Structure
- `OracleTimeStamp` Members

operator !=

This static operator determines if two `OracleTimeStamp` values are not equal.

Declaration

```csharp
// C#
public static bool operator != (OracleTimeStamp value1,
     OracleTimeStamp value2);
```

Parameters

- `value1`:
  The first `OracleTimeStam`p.

- `value2`:
  The second `OracleTimeStam`p.

Return Value

Returns `true` if two `OracleTimeStamp` values are not equal; otherwise, returns `false`.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStamp` Structure
- `OracleTimeStamp` Members
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 1-9
- 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
  The first OracleTimeStamp.
- value2
  The 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator <=
This static operator determines if the first of two OracleTimeStamp values is less than or equal to the second.
Declaration

```
// C#
public static bool operator <= (OracleTimeStamp value1,
    OracleTimeStamp value2);
```

Parameters

- `value1`  
  The first `OracleTimeStamp`.  
- `value2`  
  The 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 `OracleTimeStamps` that have a value is greater than an `OracleTimeStamps` that has a null value.  
- Two `OracleTimeStamps` that contain a null value are equal.  

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9  
- `OracleTimeStamps` Structure  
- `OracleTimeStamps` Members  

`operator -`

`operator -` subtracts the supplied value, from the supplied `OracleTimeStamps` value, and returns a new `OracleTimeStamps` structure.  

Overload List:

- `operator - (OracleTimeStamps, OracleIntervalDS)`  
  This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStamps` value, and return a new `OracleTimeStamps` structure.  
- `operator - (OracleTimeStamps, OracleIntervalYM)`  
  This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStamps` value, and returns a new `OracleTimeStamps` structure.  
- `operator - (OracleTimeStamps, TimeSpan)`  
  This static operator subtracts the supplied `TimeSpan` value, from the supplied `OracleTimeStamps` value, and returns a new `OracleTimeStamps` structure.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator - (OracleTimeStamp, OracleIntervalDS)

This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStamp value, and return a new OracleTimeStamp structure.

Declaration

```csharp
// C#
public static operator - (OracleTimeStamp value1, OracleIntervalDS value2);
```

Parameters

- `value1` An OracleTimeStamp.
- `value2` An OracleIntervalDS instance.

Return Value

An OracleTimeStamp structure.

Remarks

If either parameter has a null value, the returned OracleTimeStamp has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
// C#
public static operator - (OracleTimeStamp value1, OracleIntervalYM value2);
```

Parameters

- `value1` An OracleTimeStamp.
- `value2` An OracleIntervalYM instance.

Return Value

An OracleTimeStamp structure.
Remarks
If either parameter has a null value, the returned OracleTimeStamp has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

operator - (OracleTimeStamp, TimeSpan)
This static operator subtracts the supplied TimeSpan value, from the supplied OracleTimeStamp value, and returns a new OracleTimeStamp structure.

Declaration
// C#
public static operator - (OracleTimeStamp value1, TimeSpan value2);

Parameters
■ value1
  An OracleTimeStamp.
■ value2
  A TimeSpan instance.

Return Value
An OracleTimeStamp structure.

Remarks
If the OracleTimeStamp instance has a null value, the returned OracleTimeStamp structure has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Static Type Conversions

The OracleTimeStamp static type conversions are listed in Table 14–97.

<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 1-9
- 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

explicit operator OracleTimeStamp(OracleTimeStampLTZ)

This static type conversion operator converts an OracleTimeStampLTZ value to an OracleTimeStamp structure.

Declaration

```csharp
// C#
public static explicit operator OracleTimeStamp(OracleTimeStampLTZ value1);
```
Parameters

- value1
  
  An OracleTimeStampLTZ instance.

Return Value

The returned OracleTimeStamp contains the date and time of the OracleTimeStampLTZ structure.

Remarks

If the OracleTimeStampLTZ structure has a null value, the returned OracleTimeStamp structure also has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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

An 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 the Oracle 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 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());
    }
}
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
- 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

**implicit operator OracleTimeStamp(OracleDate)**

This static type conversion operator converts an OracleDate value to an OracleTimeStamp structure.

**Declaration**

```csharp
// C#
public static implicit operator OracleTimeStamp (OracleDate value1);
```

**Parameters**

- `value1`  
  An OracleDate instance.

**Return Value**

An OracleTimeStamp structure that contains the date and time of the OracleDate structure, `value1`.

**Remarks**

If the OracleDate structure has a null value, the returned OracleTimeStamp structure also has a null value.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

implicit operator OracleTimeStamp(DateTime)
This static type conversion operator converts a DateTime value to an OracleTimeStamp structure.

Declaration
// C#  
public static implicit operator OracleTimeStamp(DateTime value);

Parameters
- value
  A DateTime instance.

Return Value
An OracleTimeStamp structure.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

explicit operator DateTime
This static type conversion operator converts an OracleTimeStamp value to a DateTime structure.

Declaration
// C#  
public static explicit operator DateTime(OracleTimeStamp value1);

Parameters
- value1
  An OracleTimeStamp instance.

Return Value
A DateTime containing the date and time in the current instance.

Exceptions
OracleNullValueException - The OracleTimeStamp structure has a null value.

Remarks
The precision of the OracleTimeStamp can be lost during the conversion.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Properties

The OracleTimeStamp properties are listed in Table 14–98.

<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 or not 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

BinData

This property returns an array of bytes that represents an Oracle TIMESTAMP in Oracle internal format.

Declaration

\[
// C#
public byte[] BinData {get;}
\]

Property Value

A byte array that represents an Oracle TIMESTAMP in an internal format.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
Day

This property specifies the day component of an OracleTimeStamp.

Declaration

```csharp
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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```csharp
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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

Hour

This property specifies the hour component of an OracleTimeStamp.

Declaration

```csharp
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 Properties

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

Millisecond

This property gets the millisecond component of an OracleTimeStamp.

Declaration

```csharp
// 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

Minute

This property gets the minute component of an OracleTimeStamp.

Declaration

```csharp
// 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

Month

This property gets the month component of an OracleTimeStamp.

Declaration

```csharp
// 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 1-9
- 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 1-9
- 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Properties

Value

This property specifies the date and time that is stored in the OracleTimeStamp structure.

Declaration

// C#
public DateTime Value{get;}

Property Value

A DateTime.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

Year

This property gets the year component of an OracleTimeStamp.

Declaration

// C#
public int Year{get;}

Property Value

A number that represents a year. The range of Year is (-4712 to 9999).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
OracleTimeStamp Methods

The OracleTimeStamp methods are listed in Table 14–99.

Table 14–99 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>AddMillis</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 or not 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 System.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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

AddDays

This method adds the supplied number of days to the current instance.
**OracleTimeStamp Methods**

**Declaration**

```csharp
// C#
public OracleTimeStamp AddDays(double days);
```

**Parameters**

- `days`
  
  The supplied number of days. Range is \((-1,000,000,000 < days < 1,000,000,000)\)

**Return Value**

An `OracleTimeStamp`.

**Exceptions**

- `ArgumentOutOfRangeException` - The argument value is out of the specified range.
- `OracleNullValueException` - The current instance has a null value.

**AddHours**

This method adds the supplied number of hours to the current instance.

**Declaration**

```csharp
// C#
public OracleTimeStamp AddHours(double hours);
```

**Parameters**

- `hours`
  
  The supplied number of hours. Range is \((-24,000,000,000 < hours < 24,000,000,000)\).

**Return Value**

An `OracleTimeStamp`.

**Exceptions**

- `ArgumentOutOfRangeException` - The argument value is out of the specified range.
- `OracleNullValueException` - The current instance has a null value.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`
AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

```csharp
// 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 1-9
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

```csharp
// 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

```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.

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

Declaration

```c#
public OracleTimeStamp AddNanoseconds(long nanoseconds);
```

Parameters

- `nanoseconds`

  The supplied number of nanoseconds.

Return Value

An `OracleTimeStamp`.

Exceptions

- `OracleNullValueException` - The current instance has a null value.
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 \((-8.64 \times 10^{13} < seconds < 8.64 \times 10^{13})\).

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 1-9
- `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 \((-999,999,999 \leq years \leq 999,999,999)\)

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

```csharp
// 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 `OracleTimeStamp` instances. 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 `OracleTimeStamp` instances that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStamp Structure`
- `OracleTimeStamp Members`
Equals

Overrides Object

This method determines whether or not an object has the same date and time as the current OracleTimeStamp instance.

Declaration

```csharp
// 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

GetHashCode

Overrides Object

This method returns a hash code for the OracleTimeStamp 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
GetDaysBetween

This method subtracts an OracleTimeStamp value from the current instance and returns an OracleIntervalDS that represents the time difference between the supplied OracleTimeStamp structure and the current instance.

Declaration

// C#
public OracleIntervalDS GetDaysBetween(OracleTimeStamp value1);

Parameters

- value1
  The OracleTimeStamp value being subtracted.

Return Value

An OracleIntervalDS that represents the interval between two OracleTimeStamp values.

Remarks

If either the current instance or the parameter has a null value, the returned OracleIntervalDS has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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.
OracleTimeStamp Structure

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members

ToOracleDate

This method converts the current OracleTimeStamp structure to an OracleDate structure.

Declaration

// C#
public OracleDate ToOracleDate();

Return Value
The returned OracleDate contains the date and time in the current instance.

Remarks
The precision of the OracleTimeStamp value can be lost during the conversion.
If the value of the OracleTimeStamp has a null value, the value of the returned OracleDate structure has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

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();
        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 1-9
- OracleTimeStamp Structure
- OracleTimeStamp Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
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
System.Object
   System.ValueType
      Oracle.DataAccess.Types.OracleTimeStampLTZ

Declaration
// ADO.NET 2.0: C#
public struct OracleTimeStampLTZ : IComparable, INullable, IXmlSerializable

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;

        OracleIntervalDS idsDiff = tsLocal2.GetDaysBetween(tsLocal1);
        Console.WriteLine("idsDiff.Nanoseconds = "+ idsDiff.Nanoseconds);

        int nanoDiff = 0;
        while (tsLocal2 > tsLocal1)
        {
            nanoDiff += 10;
            tsLocal1 = tsLocal1.AddNanoseconds(10);
        }
        Console.WriteLine("nanoDiff = " + nanoDiff);
    }
}
Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Members
- OracleTimeStampLTZ Constructors
- OracleTimeStampLTZ Static Fields
- OracleTimeStampLTZ Static Methods
- OracleTimeStampLTZ Static 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 14–100

Table 14–100 OracleTimeStampLTZ Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleTimeStampLTZ</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 14–101.

Table 14–101 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 14–102.

Table 14–102 OracleTimeStampLTZ Static Methods

<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>
</tbody>
</table>
OracleTimeStampLTZ Static Operators

The OracleTimeStampLTZ static operators are listed in Table 14–103.

Table 14–103  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 14–104.

Table 14–104  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 Members

Table 14–104 (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 14–105.

Table 14–105 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 or not 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 14–106.

Table 14–106 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>
### OracleTimeStampLTZ Methods (Cont.)

<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 or not 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 System.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 1-9
- 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)
  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.
- OracleTimeStampLTZ(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 1-9
- 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 Structure

Parameters

- \( dt \)
  The supplied DateTime value.

Exceptions

ArgumentException - The \( dt \) parameter cannot be used to construct a valid OracleTimeStampLTZ.

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

```csharp
// 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 the Oracle 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.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());

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
- Oracle Database SQL Reference for further information on date format elements

OracleTimeStampLTZ(int, int, int)

This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date using year, month, and day.

Declaration
    // C#
    public OracleTimeStampLTZ(int year, int month, int day);

Parameters
- year
  The year provided. Range of year is (-4712 to 9999).
- month
  The month provided. Range of month is (1 to 12).
- day
  The day provided. Range of day is (1 to 31).

Exceptions
    ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.
    ArgumentException - The argument values of the parameters cannot be used to construct a valid OracleTimeStampLTZ (that is, the day is out of range for the month).
OracleTimeStampLTZ(int, int, int, int, int, int)

This constructor creates a new instance of the OracleTimeStampLTZ structure and sets its value for date and time using year, month, day, hour, minute, and second.

Declaration

// C#
public OracleTimeStampLTZ (int year, int month, int day, int hour, int minute, int second);

Parameters

- **year**
  The year provided. Range of `year` is (-4712 to 9999).
- **month**
  The month provided. Range of `month` is (1 to 12).
- **day**
  The day provided. Range of `day` is (1 to 31).
- **hour**
  The hour provided. Range of `hour` is (0 to 23).
- **minute**
  The minute provided. Range of `minute` is (0 to 59).
- **second**
  The second provided. Range of `second` is (0 to 59).

Exceptions

ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.

ArgumentException - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

OracleTimeStampLTZ(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

```csharp
// C#
public OracleTimeStampLTZ(int year, int month, int day, int hour, int minute, int second, double millisecond);
```

Parameters

- **year**
  The year provided. Range of `year` is (-4712 to 9999).

- **month**
  The month provided. Range of `month` is (1 to 12).

- **day**
  The day provided. Range of `day` is (1 to 31).

- **hour**
  The hour provided. Range of `hour` is (0 to 23).

- **minute**
  The minute provided. Range of `minute` is (0 to 59).

- **second**
  The second provided. Range of `second` is (0 to 59).

- **milliSeconds**
  The milliseconds provided. Range of `millisecond` is (0 to 999.999999).

Exceptions

- `ArgumentOutOfRangeException` - The argument value for one or more of the parameters is out of the specified range.

- `ArgumentException` - The argument values of the parameters cannot be used to construct a valid `OracleTimeStampLTZ` (that is, the day is out of range for the month).

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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

```csharp
// C#
public OracleTimeStampLTZ (int year, int month, int day, int hour, int minute, int second, int nanosecond);
```
Parameters

- **year**
  The year provided. Range of year is (-4712 to 9999).

- **month**
  The month provided. Range of month is (1 to 12).

- **day**
  The day provided. Range of day is (1 to 31).

- **hour**
  The hour provided. Range of hour is (0 to 23).

- **minute**
  The minute provided. Range of minute is (0 to 59).

- **second**
  The second provided. Range of second is (0 to 59).

- **nanosecond**
  The nanosecond provided. Range of nanosecond is (0 to 999999999).

Exceptions

- **ArgumentOutOfRangeException** - The argument value for one or more of the parameters is out of the specified range.

- **ArgumentException** - The argument values of the parameters cannot be used to construct a valid OracleTimeStampLTZ (that is, the day is out of range for the month).

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Static Fields

The OracleTimeStampLTZ static fields are listed in Table 14–107.

Table 14–107  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>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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

```csharp
// 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 1-9
- 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

```csharp
// 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Static Methods

The OracleTimeStampLTZ static methods are listed in Table 14–108.

Table 14–108 OracleTimeStampLTZ Static Methods

<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 1-9
- 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**
  - The first OracleTimeStampLTZ.
- **value2**
  - The second OracleTimeStampLTZ.
OracleTimeStampLTZ Static Methods

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 1-9
- 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 1-9
- 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
GetSysDate

This static method gets an `OracleTimeStampLTZ` structure that represents the current date and time.

**Declaration**

```csharp
// 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 1-9
- `OracleTimeStampLTZ` Structure
- `OracleTimeStampLTZ` Members

GreaterThan

This static method determines if the first of two `OracleTimeStampLTZ` values is greater than the second.

**Declaration**

```csharp
// C#
public static bool GreaterThan(OracleTimeStampLTZ value1, OracleTimeStampLTZ value2);
```

**Parameters**

- `value1`
  
  The first `OracleTimeStampLTZ`.

- `value2`
  
  The 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 1-9
- `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
  The first OracleTimeStampLTZ.
- value2
  The 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 1-9
- 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
  The first OracleTimeStampLTZ.
- value2
  The 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 OracleTimeStampLTZs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

LessThanOrEqual
This static method determines if the first of two OracleTimeStampLTZ values is less than or equal to the second.

Declaration

// C#
public static bool LessThanOrEqual(OracleTimeStampLTZ value1, OracleTimeStampLTZ value2);

Parameters

- value1
  The first OracleTimeStampLTZ.
- value2
  The 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 OracleTimeStampLTZs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
  The first OracleTimeStampLTZ.
- value2
  The 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 1-9
- 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 the Oracle NLS_TIMESTAMP_FORMAT parameter.

ArgumentException - 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.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 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 1-9
- `OracleTimeStampLTZ Structure`
- `OracleTimeStampLTZ Members`
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

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**

ArgumentException - 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Static Operators

The OracleTimeStampLTZ static operators are listed in Table 14–109.

Table 14–109  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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator+

operator+ adds the supplied value to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.

Overload List:
- operator + (OracleTimeStampLTZ, OracleIntervalDS)
  This static operator adds the supplied OracleIntervalDS to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.
- operator + (OracleTimeStampLTZ, OracleIntervalYM)
  This static operator adds the supplied OracleIntervalYM to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.
- operator + (OracleTimeStampLTZ, TimeSpan)
  This static operator adds the supplied TimeSpan to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator + (OracleTimeStampLTZ, OracleIntervalDS)

This static operator adds the supplied OracleIntervalDS to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.

Declaration

// C#
public static operator +(OracleTimeStampLTZ value1, OracleIntervalDS value2);

Parameters

- **value1**
  - An OracleTimeStampLTZ.
- **value2**
  - An OracleIntervalDS.

Return Value

An OracleTimeStampLTZ.

Remarks

If either parameter has a null value, the returned OracleTimeStampLTZ has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator + (OracleTimeStampLTZ, OracleIntervalYM)

This static operator adds the supplied OracleIntervalYM to the supplied OracleTimeStampLTZ and returns a new OracleTimeStampLTZ structure.

Declaration

// C#
public static operator +(OracleTimeStampLTZ value1, OracleIntervalYM value2);

Parameters

- **value1**
  - An OracleTimeStampLTZ.
- **value2**
  - An OracleIntervalYM.
Return Value
An OracleTimeStampLTZ.

Remarks
If either parameter has a null value, the returned OracleTimeStampLTZ has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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);
OracleTimeStampLTZ Static Operators

Parameters
- **value1**
  The first OracleTimeStampLTZ.
- **value2**
  The 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 OracleTimeStampLTZs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator >
This static operator determines if the first of two OracleTimeStampLTZ values is greater than the second.

Declaration
```csharp
// C#
public static bool operator > (OracleTimeStampLTZ value1,
                               OracleTimeStampLTZ value2);
```

Parameters
- **value1**
  The first OracleTimeStampLTZ.
- **value2**
  The 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 OracleTimeStampLTZs that contain a null value are equal.
**operator &gt;=**

This static operator determines if the first of two `OracleTimeStampLTZ` values is greater than or equal to the second.

**Declaration**

// C#
public static bool operator &gt;= (OracleTimeStampLTZ value1, OracleTimeStampLTZ value2);

**Parameters**

- **value1**
  - An `OracleTimeStampLTZ`.
- **value2**
  - The 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 1-9
- `OracleTimeStampLTZ Structure`
- `OracleTimeStampLTZ Members`

**operator !=**

This static operator determines if two `OracleTimeStampLTZ` values are not equal.

**Declaration**

// C#
public static bool operator != (OracleTimeStampLTZ value1, OracleTimeStampLTZ value2);

**Parameters**

- **value1**
  - The first `OracleTimeStampLTZ`.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStampLTZ Structure`
- `OracleTimeStampLTZ Members`
The second `OracleTimeStampLTZ`.

Return Value

Returns `true` if two `OracleTimeStampLTZ` values are not equal; otherwise returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStampLTZ` Structure
- `OracleTimeStampLTZ` Members

### 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`
  
  The first `OracleTimeStampLTZ`.

- `value2`
  
  The second `OracleTimeStampLTZ`.

Return Value

Returns `true` if the first `OracleTimeStampLTZ` is less than the second; otherwise, returns `false`.

Remarks

The following rules apply to the behavior of this method.

- Any `OracleTimeStampLTZ` that has a value is greater than an `OracleTimeStampLTZ` that has a null value.
- Two `OracleTimeStampLTZ`s that contain a null value are equal.
operator <=

This static operator determines if the first of two OracleTimeStampLTZ values is less than or equal to the second.

Declaration

// C#
public static bool operator <= (OracleTimeStampLTZ value1,
                                 OracleTimeStampLTZ value2);

Parameters

- value1
  The first OracleTimeStampLTZ.
- value2
  The second OracleTimeStampLTZ.

Return Value

Returns true if the first OracleTimeStampLTZ is less than or equal to the second; otherwise, returns false.

Remarks

The following rules apply to the behavior of this method.

- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZ that contain a null value are equal.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

operator -

operator - subtracts the supplied value, from the supplied OracleTimeStampLTZ value, and returns a new OracleTimeStampLTZ structure.

Overload List:

- operator - (OracleTimeStampLTZ, OracleIntervalDS)
  This static operator subtracts the supplied OracleIntervalDS value, from the supplied OracleTimeStampLTZ value, and return a new OracleTimeStampLTZ structure.
- operator - (OracleTimeStampLTZ, OracleIntervalYM)
This static operator subtracts the supplied `OracleIntervalYM` value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

- **operator - (OracleTimeStampLTZ, TimeSpan)**
  
  This static operator subtracts the supplied `TimeSpan` value, from the supplied `OracleTimeStampLTZ` value, and returns a new `OracleTimeStampLTZ` structure.

  **Declaration**
  
  ```csharp
  // C#
  public static operator - (OracleTimeStampLTZ value1, TimeSpan 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 1-9
  
  - `OracleTimeStampLTZ Structure`
  
  - `OracleTimeStampLTZ Members`

- **operator - (OracleTimeStampLTZ, OracleIntervalDS)**
  
  This static operator subtracts the supplied `OracleIntervalDS` value, from the supplied `OracleTimeStampLTZ` value, and return a new `OracleTimeStampLTZ` structure.

  **Declaration**
  
  ```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 1-9
  
  - `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#
  ```
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 1-9
- 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Static Type Conversions

The OracleTimeStampLTZ static type conversions are listed in Table 14–110.

Table 14–110  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>
<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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ

This static type conversion operator 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ(OracleTimeStamp)

This static type conversion operator converts an OracleTimeStamp value to an OracleTimeStampLTZ structure.

Declaration

// C#
public static explicit operator OracleTimeStampLTZ (OracleTimeStamp value1);
Parameters
- value1
  An OracleTimeStamp.

Return Value
The OracleTimeStampLTZ structure contains the date and time of the OracleTimeStampTZ structure.

Remarks
If the OracleTimeStamp structure has a null value, the returned OracleTimeStampLTZ structure also has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ(OracleTimeStampTZ)
This static type conversion operator converts an OracleTimeStampTZ value to an OracleTimeStampLTZ structure.

Declaration
// C#
public static explicit operator OracleTimeStampLTZ
(OracleTimeStampTZ value1);

Parameters
- value1
  An OracleTimeStampTZ instance.

Return Value
The OracleTimeStampLTZ structure contains the date and time in the OracleTimeStampTZ structure (which is normalized to the client local time zone).

Remarks
If the OracleTimeStampTZ structure has a null value, the returned OracleTimeStampLTZ structure also has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

explicit operator OracleTimeStampLTZ(string)
This static type conversion operator converts the supplied string to an OracleTimeStampLTZ structure.
OracleTimeStampLTZ Static Type Conversions

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 the Oracle 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());
    }
}
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
- Oracle Database SQL Reference for further information on datetime format elements

**implicit operator OracleTimeStampLTZ**

**Declaration**

```csharp
public static implicit operator OracleTimeStampLTZ(OracleDate value1);
```

**Parameters**
- `value1` An `OracleDate`.

**Return Value**
The returned `OracleTimeStampLTZ` structure contains the date and time in the `OracleDate` structure.

**Remarks**
If the `OracleDate` structure has a null value, the returned `OracleTimeStampLTZ` structure also has a null value.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
■ OracleTimeStampLTZ Structure
■ OracleTimeStampLTZ Members
OracleTimeStampLTZ Properties

The OracleTimeStampLTZ properties are listed in Table 14–111.

Table 14–111 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 or not 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

BinData

This property returns an array of bytes that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE in Oracle internal format.

Declaration

// C#
public byte[] BinData {get;}

Property Value

A byte array that represents an Oracle TIMESTAMP WITH LOCAL TIME ZONE internal format.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
Day

This property specifies the day component of an OracleTimeStampLTZ.

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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

IsNull

This property indicates whether or not the current instance has a null value.

Declaration

```csharp
// 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Hour

This property specifies the hour component of an OracleTimeStampLTZ.

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.
OracleTimeStampLTZ Properties

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Minute

This property gets the minute component of an OracleTimeStampLTZ.

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

Property Value
A number that represent a minute. Range of Minute is (0 to 59).

Exceptions
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Month

This property gets the month component of an OracleTimeStampLTZ.

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ 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 1-9
- 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 1-9
- 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Properties

Value

This property specifies the date and time that is stored in the OracleTimeStampLTZ structure.

Declaration

// C#
public DateTime Value{get;}

Property Value

A DateTime.

Exceptions

OracleNullValueException - The current instance has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Year

This property gets the year component of an OracleTimeStampLTZ.

Declaration

// C#
public int Year{get;}

Property Value

A number that represents a year. The range of Year is (-4712 to 9999).

Exceptions

OracleNullValueException - The current instance has a null value.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
OracleTimeStampLTZ Methods

The OracleTimeStampLTZ methods are listed in Table 14–112.

<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 or not 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 System.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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
AddDays

This method adds the supplied number of days to the current instance.

Declaration

// C#
public OracleTimeStampLTZ AddDays(double days);

Parameters

■ days

The supplied number of days. Range is (-1,000,000,000 < days < 1,000,000,000)

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.
ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ OracleTimeStampLTZ Structure
■ OracleTimeStampLTZ Members

AddHours

This method adds the supplied number of hours to the current instance.

Declaration

// C#
public OracleTimeStampLTZ AddHours(double hours);

Parameters

■ hours

The supplied number of hours. Range is (-24,000,000,000 < hours < 24,000,000,000).

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.
ArgumentOutOfRangeException - The argument value is out of the specified range.
AddMilliseconds

This method adds the supplied number of milliseconds to the current instance.

Declaration

// C#
public OracleTimeStampLTZ AddMilliseconds(double milliseconds);

Parameters

- milliseconds
  The supplied number of milliseconds. Range is \((-8.64 \times 10^{16} < \text{milliseconds} < 8.64 \times 10^{16})\).

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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddMinutes

This method adds the supplied number of minutes to the current instance.

Declaration

// C#
public OracleTimeStampLTZ AddMinutes(double minutes);

Parameters

- minutes
  The supplied number of minutes. Range is \((-1,440,000,000,000 < \text{minutes} < 1,440,000,000,000)\).

Return Value

An OracleTimeStampLTZ.

Exceptions

OracleNullValueException - The current instance has a null value.
OracleTimeStampLTZ Methods

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

AddNanoseconds

This method adds the supplied number of nanoseconds to the current instance.

Declaration

// C#
public OracleTimeStampLTZ AddNanoseconds(long nanoseconds);

Parameters

- **nanoseconds**
  
  The supplied number of nanoseconds.

Return Value

An OracleTimeStampLTZ.
Exceptions

OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- 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.
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

Equals

Overrides Object
This method determines whether or not an object has the same date and time as the current OracleTimeStampLTZ instance.

Declaration
// C#
public override bool Equals(object obj);

Parameters
- obj
  The object being compared to the current OracleTimeStampLTZ instance.

Return Value
Returns true if the obj is of type OracleTimeStampLTZ and represents the same date and time; otherwise, returns false.

Remarks
The following rules apply to the behavior of this method.
- Any OracleTimeStampLTZ that has a value is greater than an OracleTimeStampLTZ that has a null value.
- Two OracleTimeStampLTZs that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GetHashCode

Overrides Object
This method returns a hash code for the OracleTimeStampLTZ instance.

Declaration
// C#
public override int GetHashCode();

Return Value
A number that represents the hash code.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

GetDaysBetween

This method subtracts an OracleTimeStampLTZ value from the current instance and returns an OracleIntervalDS that represents the difference.

Declaration

```csharp
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 1-9
- 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

```csharp
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 1-9
- 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 1-9
- 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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members

ToString

Overrides Object

This method converts the current OracleTimeStampLTZ structure to a string.

Declaration

// C#
public override string ToString();

Return Value

A string that represents the same date and time as the current OracleTimeStampLTZ structure.

Remarks

The returned value is a string representation of the OracleTimeStampLTZ in the format specified by the OracleGlobalization.TimeStampFormat property of the thread.

The names and abbreviations used for months and days are in the language specified by the DateLanguage and Calendar properties of the thread's
OracleGlobalization object. If any of the thread’s globalization properties are set to null or an empty string, the client computer’s settings are used.

Example

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

class ToStringSample
{
    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 1-9
- OracleTimeStampLTZ Structure
- OracleTimeStampLTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

ToUniversalTime

This method converts the current local time to Coordinated Universal Time (UTC).

Declaration

```csharp
public OracleTimeStampLTZ ToUniversalTime();
```

Return Value

An OracleTimeStampLTZ structure.

Remarks

If 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 1-9
- 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
System.Object
   System.ValueType
       Oracle.DataAccess.Types.OracleTimeStampTZ

Declaration
// ADO.NET 2.0: C#
public struct OracleTimeStampTZ : IComparable, INullable, IXmlSerializable

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 tisz1=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 tisz2 = ts.ToOracleTimeStampTZ();

        // Calculate the difference between tisz1 and tisz2
        OracleIntervalDS idsDiff = tisz1.GetDaysBetween(tisz2);

        // Display information
        Console.WriteLine("tisz1.TimeZone = " + tisz1.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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace“ on page 1-9
- 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 14–113.

Table 14–113  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 14–114.

Table 14–114  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 14–115.

Table 14–115  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 Members

Table 14–115 (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 14–116.

Table 14–116 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 14–117.

Table 14–117 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</td>
</tr>
</tbody>
</table>
OracleTimeStampTZ Properties

The OracleTimeStampTZ properties are listed in Table 14-118.

**Table 14-118  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 or not 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 14-119.

**Table 14-119  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 14–119 (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 or not 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 System.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 a 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 1-9
- 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)**
  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, string)**
  This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using year, month, day, hour, minute, second, and time zone data.

- **OracleTimeStampTZ(int, int, int, int, int, int, double)**
  This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using year, month, day, hour, minute, second, and millisecond.

- **OracleTimeStampTZ(int, int, int, int, int, int, double, string)**
  This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using year, month, day, hour, minute, second, millisecond, and time zone data.

- **OracleTimeStampTZ(int, int, int, int, int, int, int)**
  This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using year, month, day, hour, minute, second, and nanosecond.

- **OracleTimeStampTZ(int, int, int, int, int, int, int, string)**
This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value for date and time using year, month, day, hour, minute, second, nanosecond, and time zone data.

- **OracleTimeStampTZ(byte[])**
  This constructor creates a new instance of the `OracleTimeStampTZ` structure and sets its value to the provided byte array, that represents the internal Oracle `TIMESTAMP WITH TIME ZONE` format.

- **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 1-9
  - `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 1-9
  - `OracleTimeStampTZ` Structure
  - `OracleTimeStampTZ` Members
**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 1-9
- 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133
- Oracle Database SQL Reference for further information on date format elements

OracleTimeStampTZ(int, int, int)

This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value for date and time using year, month, and day.

Declaration
// C#
public OracleTimeStampTZ(int year, int month, int day);

Parameters
- **year**
  The year provided. Range of **year** is (-4712 to 9999).
- **month**
  The month provided. Range of **month** is (1 to 12).
- **day**
  The day provided. Range of **day** is (1 to 31).
Exceptions

ArgumentOutOfRangeException - The argument value for one or more of the parameters is out of the specified range.

ArgumentException - The argument values of the parameters cannot be used to construct a valid OracleTimeStampTZ (that is, the day is out of range for the month).

Remarks

The time zone is set to the OracleGlobalization.TimeZone of the thread.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 Constructors

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.

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.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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

```csharp
// 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

- 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 timeZone is null, the OracleGlobalization.TimeZone of the thread is used.
OracleTimeStampTZ Constructors

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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

OracleTimeSpanTZ(int, int, int, int, int, int, double)
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 1-9
- `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.
OracleDataTimeStampTZ Structure

OracleDataTimeStampTZ(int, int, int, int, int, int, int, string)
This constructor creates a new instance of the OracleDataTimeStampTZ structure and sets its value for date and time using year, month, day, hour, minute, second, nanosecond, and time zone data.

Declaration
// C#
public OracleDataTimeStampTZ(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 OracleDataTimeStampTZ (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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

**OracleTimeStampTZ(byte [])**

This constructor creates a new instance of the OracleTimeStampTZ structure and sets its value to the provided byte array, that represents the internal Oracle TIMESTAMP WITH TIME ZONE format.

**Declaration**

// C#
public OracleTimeStampLTZ (byte[] bytes);

**Parameters**

- **bytes**

  The provided byte array that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format.

**Exceptions**

ArgumentException - bytes is not in internal Oracle TIMESTAMP WITH TIME ZONE format or bytes is not a valid Oracle TIMESTAMP WITH TIME ZONE.

ArgumentNullException - bytes is null.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
OracleTimeStampTZ Static Fields

The OracleTimeStampTZ static fields are listed in Table 14–120.

<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 1-9
- 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
public static readonly OracleTimeStampTZ MaxValue;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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
public static readonly OracleTimeStampTZ MinValue;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`
OracleTimeStampTZ Static Methods

The OracleTimeStampTZ static methods are listed in Table 14–121.

Table 14–121  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 1-9
- 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`
The first OracleTimeStampTZ.
- `value2`
The 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 1-9
- `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 1-9
- `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`
  The first `OracleTimeStampTZ`.
- `value2`
  The 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 1-9
- 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
  The first OracleTimeStampTZ.

■ value2
  The 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 1-9
- 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
   The first OracleTimeStampTZ.
■ value2
   The second OracleTimeStampTZ.

Return Value
Returns true if the first of two OracleTimeStampTZ values is less than the second. Returns false otherwise.

Remarks
The following rules apply to the behavior of this method.

■ Any OracleTimeStampTZ that has a value is greater than an OracleTimeStampTZ that has a null value.
■ Two OracleTimeStampTZs that contain a null value are equal.

See Also:
■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ 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
   The first OracleTimeStampTZ.
■ value2
   The second OracleTimeStampTZ.

Return Value
Returns true if the first of two OracleTimeStampTZ values is less than or equal to the second. Returns false otherwise.

Remarks
The following rules apply to the behavior of this method.
■ Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.

■ Two `OracleTimeStampTZs` that contain a null value are equal.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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**
  The first `OracleTimeStampTZ`.

- **value2**
  The 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 1-9
- `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 the Oracle **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
// C#
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());
    }
}
```
See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
OracleTimeStampTZ Static Operators

The OracleTimeStampTZ static operators are listed in Table 14–122.

<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 1-9
- 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.
operator +(OracleTimeStampTZ, OracleIntervalDS)

This static operator adds the supplied OracleIntervalDS to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure.

Declaration

// C#
public static operator +(OracleTimeStampTZ value1,
                        OracleIntervalDS value2);

Parameters

■ value1
  An OracleTimeStampTZ.

■ value2
  An OracleIntervalDS.

Return Value

An OracleTimeStampTZ.

Remarks

If either parameter has a null value, the returned OracleTimeStampTZ has a null value.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ OracleTimeStampTZ Structure
■ OracleTimeStampTZ Members

operator +(OracleTimeStampTZ, OracleIntervalYM)

This static operator adds the supplied OracleIntervalYM to the supplied OracleTimeStampTZ and returns a new OracleTimeStampTZ structure.

Declaration

// C#
public static operator +(OracleTimeStampTZ value1,
                        OracleIntervalYM value2);

Parameters

■ value1
  An OracleTimeStampTZ.

■ value2
  An OracleIntervalYM.
### Return Value
An `OracleTimeStampTZ`.

### Remarks
If either parameter has a null value, the returned `OracleTimeStampTZ` has a null value.

#### See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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
```csharp
// 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 1-9
- `OracleTimeStampTZ` Structure
- `OracleTimeStampTZ` Members

### `operator ==`
This static operator determines if two `OracleTimeStampTZ` values are equal.

#### Declaration
```csharp
// C#
public static bool operator ==(OracleTimeStampTZ value1, OracleTimeStampTZ value2);
```

#### Parameters
- `value1`

- `value2`
The first `OracleTimeStampTZ`.

- `value2`
  The second `OracleTimeStampTZ`.

**Return Value**
Returns `true` if they are equal; otherwise returns `false`.

**Remarks**
The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`

**operator >**
This static operator determines if the first of two `OracleTimeStampTZ` values is greater than the second.

**Declaration**

```csharp
// C#
public static bool operator > (OracleTimeStampTZ value1,
                     OracleTimeStampTZ value2);
```

**Parameters**

- `value1`
  The first `OracleTimeStampTZ`.
- `value2`
  The second `OracleTimeStampTZ`.

**Return Value**
Returns `true` if the first `OracleTimeStampTZ` value is greater than the second; otherwise, returns `false`.

**Remarks**
The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.
operator >=

This static operator determines if the first of two OracleTimeStampTZ values is greater than or equal to the second.

**Declaration**

```csharp
// C#
public static bool operator >= (OracleTimeStampTZ value1, OracleTimeStampTZ value2);
```

**Parameters**

- `value1`
  The first OracleTimeStampTZ.

- `value2`
  The 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

operator !=

This static operator determines if two OracleTimeStampTZ values are not equal.

**Declaration**

```csharp
// C#
public static bool operator != (OracleTimeStampTZ value1, OracleTimeStampTZ value2);
```

**Parameters**

- `value1`
  The first OracleTimeStampTZ.
value2

The 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 1-9
- 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
  
The first OracleTimeStampTZ.
- value2
  
The 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**

```csharp
// C#
public static bool operator <= (OracleTimeStampTZ value1,
    OracleTimeStampTZ value2);
```

**Parameters**

- `value1`  
  The first `OracleTimeStampTZ`.

- `value2`  
  The second `OracleTimeStampTZ`.

**Return Value**

Returns `true` if the first `OracleTimeStampTZ` is less than or equal to the second; otherwise, returns `false`.

**Remarks**

The following rules apply to the behavior of this method.

- Any `OracleTimeStampTZ` that has a value is greater than an `OracleTimeStampTZ` that has a null value.
- Two `OracleTimeStampTZ`s that contain a null value are equal.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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 1-9
■ 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 1-9
■ 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 Static Operators

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 1-9
- 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
OracleTimeStampTZ Static Type Conversions

The OracleTimeStampTZ static type conversions are listed in Table 14–123.

<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 1-9
- 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

explicit operator OracleTimeStampTZ(OracleTimeStamp)

This static type conversion operator converts an OracleTimeStamp value to an OracleTimeStampTZ structure.
Declaration

```csharp
// 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 1-9
- `OracleTimeStampTZ` Structure
- `OracleTimeStampTZ` Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

**explicit operator OracleTimeStampTZ(OracleTimeStampLTZ)**

This static type conversion operator converts an `OracleTimeStampLTZ` value to an `OracleTimeStampTZ` structure.

Declaration

```csharp
// 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

An OracleTimeStampTZ value.

Exceptions

ArgumentException - The tsStr is an invalid string representation of an Oracle TIMESTAMP WITH TIME ZONE, or the tsStr is not in the timestamp format specified by the thread's OracleGlobalization.TimeStampTZFormat property, which represents the Oracle 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.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);
Console.WriteLine(tstz.ToString());
}
}

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

**implicit operator OracleTimeStampTZ**

**implicit operator** OracleTimeStampTZ converts a DateTime structure to an OracleTimeStampTZ structure.

**Overload List:**
- implicit operator OracleTimeStampTZ(OracleDate)
  
  This static type conversion operator converts an OracleDate value to an OracleTimeStampTZ structure.

- implicit operator OracleTimeStampTZ(DateTime)
  
  This static type conversion operator converts a DateTime structure to an OracleTimeStampTZ structure.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

**implicit operator OracleTimeStampTZ(OracleDate)**

This static type conversion operator converts an OracleDate value to an OracleTimeStampTZ structure.

**Declaration**

```csharp
// C#
public static implicit operator OracleTimeStampTZ(OracleDate value1);
```

**Parameters**
- `value1`
An OracleDate.

**Return Value**
The returned OracleTimeStampTZ contains the date and time from the OracleDate and the time zone from the OracleGlobalization.TimeZone of the thread.

**Remarks**
The OracleGlobalization.TimeZone of the thread is used to convert from an OracleDate to an OracleTimeStampTZ structure. If the OracleDate structure has a null value, the returned OracleTimeStampTZ structure also has a null value.

**See Also:**
- "Oracle.DataAccess.Types Namespace” on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class” on page 10-2
- "Globalization Support” on page 3-133

**implicit operator OracleTimeStampTZ(DateTime)**
This static type conversion operator converts a DateTime structure to an OracleTimeStampTZ structure.

**Declaration**
// C#
public static implicit operator OracleTimeStampTZ (DateTime value1);

**Parameters**
- value1
  A DateTime structure.

**Return Value**
The returned OracleTimeStampTZ contains the date and time from the DateTime and the time zone from the OracleGlobalization.TimeZone of the thread.

**Remarks**
The OracleGlobalization.TimeZone of the thread is used to convert from a DateTime to an Oracle TimeStampTZ structure.

**See Also:**
- "Oracle.DataAccess.Types Namespace” on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class” on page 10-2
- "Globalization Support” on page 3-133
explicit operator DateTime

This static type conversion operator converts an OracleTimeStampTZ value to a
DateTime structure and truncates the time zone information.

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 1-9
■ OracleTimeStampTZ Structure
■ OracleTimeStampTZ Members
OracleTimeStampTZ Properties

The OracleTimeStampTZ properties are listed in Table 14–124.

Table 14–124  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 or not 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

BinData

This property returns an array of bytes that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format.

Declaration

// C#
public byte[] BinData {get;}

Property Value

The provided byte array that represents an Oracle TIMESTAMP WITH TIME ZONE in Oracle internal format.

Exceptions

OracleNullValueException - The current instance has a null value.
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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

IsNull

This property indicates whether or not 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 1-9
- 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

**Millissecond**
This property gets the millisecond component of an OracleTimeStampTZ in the current time zone.

**Declaration**
// C#
public double Millisecond{get;}

**Property Value**
A number that represents a millisecond. Range of **Millisecond** is (0 to 999.999999)

**Exceptions**
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

**Minute**
This property gets the minute component of an OracleTimeStampTZ in the current time zone.

**Declaration**
// C#
public int Minute{get;}

**Property Value**
A number that represent a minute. Range of **Minute** is (0 to 59).

**Exceptions**
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

Second

This property gets the second component of an OracleTimeStampTZ in the current time zone.

Declaration

```c#
public int Second { get; }
```

Property Value

A number that represents a second. Range of Second is (0 to 59).
Exceptions
OracleNullValueException - The current instance has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

TimeZone
This property returns the time zone of the OracleTimeStampTZ instance.

Declaration
// C#
public string TimeZone{get;}

Property Value
A string that represents the time zone.

Remarks
If no time zone is specified in the constructor, this property is set to the thread's OracleGlobalization.TimeZone by default.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

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 1-9
- 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
OracleTimeStampTZ Methods

The OracleTimeStampTZ methods are listed in Table 14–125.

<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 or not 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 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 System.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>
AddDays

This method adds the supplied number of days to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddDays(double days);
```

Parameters

- **days**
  
The supplied number of days. Range is (-1,000,000,000 < days < 1,000,000,000)

Return Value

An `OracleTimeStampTZ`.

Exceptions

- `OracleNullValueException` - The current instance has a null value.
- `ArgumentOutOfRangeException` - The argument value is out of the specified range.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`

AddHours

This method adds the supplied number of hours to the current instance.

Declaration

```
// C#
public OracleTimeStampTZ AddHours(double hours);
```

Parameters

- **hours**
  
The supplied number of hours. Range is (-24,000,000,000 < hours < 24,000,000,000).

Return Value

An `OracleTimeStampTZ`.

Exceptions

- `OracleNullValueException` - The current instance has a null value.
OracleTimeStampTZ Structure

ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddMilliseconds
This method adds the supplied number of milliseconds to the current instance.

Declaration
// C#
public OracleTimeStampTZ AddMilliseconds(double milliseconds);

Parameters
- milliseconds
  The supplied number of milliseconds. Range is (-8.64 * 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddMinutes
This method adds the supplied number of minutes to the current instance.

Declaration
// C#
public OracleTimeStampTZ AddMinutes(double minutes);

Parameters
- minutes
  The supplied number of minutes. Range is (-1,440,000,000,000 < minutes < 1,440,000,000,000).

Return Value
An OracleTimeStampTZ.
Exceptions
OracleNullValueException - The current instance has a null value.
ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

AddSeconds
This method adds the supplied number of seconds to the current instance.

Declaration
// C#
public OracleTimeStampTZ AddSeconds(double seconds);

Parameters
- seconds
  The supplied number of seconds. Range is (-8.64 * 1013 < seconds < 8.64 * 1013).

Return Value
An OracleTimeStampTZ.

Exceptions
OracleNullValueException - The current instance has a null value.
ArgumentOutOfRangeException - The argument value is out of the specified range.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members

CompareTo
This method compares the current OracleTimeStampTZ instance to an object, and returns an integer that represents their relative values.

Declaration
// C#
public int CompareTo(object obj);

Parameters
■ obj
   The object being compared to the current OracleTimeStampTZ instance.

Return Value
The method returns a number that is:
Less than zero: if the current OracleTimeStampTZ instance value is less than that of obj.
Zero: if the current OracleTimeStampTZ instance and obj values are equal.
Greater than zero: if the current OracleTimeStampTZ instance value is greater than that of obj.

Implements
IComparable

Exceptions
ArgumentOutOfRangeException - 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.
Two `OracleTimeStampTZs` that contain a null value are equal.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTimeStampTZ Structure`
- `OracleTimeStampTZ Members`

Equals

Overrides `Object`  
This method determines whether or not 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 1-9
- `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 1-9
- 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 1-9
- 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 1-9
- 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 1-9
■ 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 1-9
■ OracleTimeStampTZ Structure
■ OracleTimeStampTZ Members

ToOracleDate

This method converts the current OracleTimeStampTZ structure to an OracleDate structure.
OracleTimeStampTZ Methods

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 1-9
- 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 1-9
- 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 1-9
- 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 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);

// Prints "US/Pacific"
Console.WriteLine("tstz1.TimeZone = " + tstz1.TimeZone);

// Prints "US/Eastern"
Console.WriteLine("tstz2.TimeZone = " + tstz2.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 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
- "OracleGlobalization Class" on page 10-2
- "Globalization Support" on page 3-133

ToUniversalTime

This method converts the current datetime to Coordinated Universal Time (UTC).

Declaration

// C#
public OracleTimeStampTZ ToUniversalTime();

Return Value
An OracleTimeStampTZ structure.

Remarks
If the current instance has a null value, the value of the returned OracleTimeStampTZ structure has a null value.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTimeStampTZ Structure
- OracleTimeStampTZ Members
INullable Interface

The INullable interface is used to determine whether or not an ODP.NET type has a NULL value.

Declaration

C# INTERFACE
public interface INullable

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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- INullable Interface Members
- INullable Interface Properties
**INullable Interface Members**

INullable members are listed in the following tables.

**INullable Interface Properties**

INullable interface properties are listed in Table 14–126.

<table>
<thead>
<tr>
<th>Public Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the ODP.NET type has a <strong>NULL</strong> value</td>
</tr>
</tbody>
</table>

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- INullable Interface
INullable Interface Properties

INullable interface properties are listed in Table 14–126.

Table 14–127 INullable Interface Properties

<table>
<thead>
<tr>
<th>Public Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the ODP.NET type has a NULL value</td>
</tr>
</tbody>
</table>

IsNull

This property indicates whether or not the ODP.NET type has a NULL value.

Declaration

```csharp
// C#
bool IsNull {get;}
```

Property Value

Returns true if the ODP.NET type has a NULL value; otherwise, returns false.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- INullable Interface
- INullable Interface Members
This section covers the ODP.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
System.Object
    System.Exception
        System.SystemException
            Oracle.DataAccess.Types.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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTypeException Members
- OracleTypeException Constructors
- OracleTypeException Static Methods
- OracleTypeException Properties
- OracleTypeException Methods
OracleTypeException Class

OracleTypeException Members

OracleTypeException members are listed in the following tables.

OracleTypeException Constructors
The OracleTypeException constructors are listed in Table 15–1.

Table 15–1 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 15–2.

Table 15–2 OracleTypeException Static Methods

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

OracleTypeException Properties
The OracleTypeException properties are listed in Table 15–3.

Table 15–3 OracleTypeException Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from System.SystemException.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 System.SystemException.Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
</tbody>
</table>

OracleTypeException Methods
The OracleTypeException methods are listed in Table 15–4.

Table 15–4 OracleTypeException Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
</tbody>
</table>
### Table 15-4 (Cont.) OracleTypeException Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetObjectData</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.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 1-9
- 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 1-9
  - 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 1-9
  - 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 OracleTypeException (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 1-9
- OracleTypeException Class
- OracleTypeException Members
OracleTypeException Static Methods

The OracleTypeException static methods are listed in Table 15–5.

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

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTypeException Class
- OracleTypeException Members
OracleTypeException Properties

The OracleTypeException properties are listed in Table 15–6.

Table 15–6 OracleTypeException Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from System.SystemException.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 System.SystemException.Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 1-9
- 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 1-9
- OracleTypeException Class
- OracleTypeException Members
OracleTypeException Methods

The OracleTypeException methods are listed in Table 15–7.

### Table 15–7 OracleTypeException Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.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 1-9
- 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 1-9
- OracleTypeException Class
- OracleTypeException Members
OracleNullValueException Class

The OracleNullValueException represents an exception that is thrown when trying to access an ODP.NET Types structure that has a null value.

Class Inheritance
System.Object
    System.Exception
        System.SystemException
            System.OracleTypeException
                Oracle.DataAccess.Types.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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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 15–8.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleNullValueException</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 15–9.

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

OracleNullValueException Properties

The OracleNullValueException properties are listed in Table 15–10.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from System.SystemException.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 System.SystemException.Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
</tbody>
</table>

OracleNullValueException Methods

The OracleNullValueException methods are listed in Table 15–11.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
</tbody>
</table>
Table 15–11 (Cont.) OracleNullValueException Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from OracleTypeException</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- 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, errMsg.

**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 1-9
- OracleNullValueException Class
- OracleNullValueException Members

**OracleNullValueException(string)**

This constructor creates a new instance of the OracleNullValueException class with the specified error message, errMsg.

**Declaration**

// C#
public OracleNullValueException (string errMsg);

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleNullValueException Class
- OracleNullValueException Members

**Parameters**

- **errMsg**
  The specified error message.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleNullValueException Class
- OracleNullValueException Members
OracleNullValueException Static Methods

The OracleNullValueException static methods are listed in Table 15–12.

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleNullValueException Class
- OracleNullValueException Members
OracleNullValueException Properties

The OracleNullValueException properties are listed in Table 15–13.

Table 15–13 OracleNullValueException Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from System.SystemException.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 System.SystemException.Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleNullValueException Class
- OracleNullValueException Members
The OracleNullValueException methods are listed in Table 15–14.

**Table 15–14  OracleNullValueException Methods**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from OracleTypeException</td>
</tr>
</tbody>
</table>
The `OracleTruncateException` class represents an exception that is thrown when truncation in a ODP.NET Types class occurs.

**Class Inheritance**

```
System.Object
    System.Exception
        System.SystemException
            System.OracleTypeException
                Oracle.DataAccess.Types.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`
- **ODP.NET Version:** ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `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 15–15.

Table 15–15  OracleTruncateException Constructors

<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 15–16.

Table 15–16  OracleTruncateException Static Methods

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

OracleTruncateException Properties
The OracleTruncateException properties are listed in Table 15–17.

Table 15–17  OracleTruncateException Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from System.SystemException.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 System.SystemException.Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
</tbody>
</table>

OracleTruncateException Methods
The OracleTruncateException methods are listed in Table 15–18.

Table 15–18  OracleTruncateException Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
</tbody>
</table>
### Table 15–18  (Cont.) OracleTruncateException Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from OracleTypeException</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTruncateException Class
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 1-9
- `OracleTruncateException` Class
- `OracleTruncateException Members`

### `OracleTruncateException()`
This constructor creates a new instance of the `OracleTruncateException` class with its default properties.

**Declaration**
```csharp
// C#
public OracleTruncateException();
```

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTruncateException` Class
- `OracleTruncateException Members`

### `OracleTruncateException(string)`
This constructor creates a new instance of the `OracleTruncateException` class with the specified error message, `errMessage`.

**Declaration**
```csharp
// C#
public OracleTruncateException (string errMessage);
```

**Parameters**
- `errMessage`
  The specified error message.
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTruncateException Class
- OracleTruncateException Members
The `OracleTruncateException` static methods are listed in Table 15–19.

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

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTruncateException Class`
- `OracleTruncateException Members`
OracleTruncateException Properties

The OracleTruncateException properties are listed in Table 15–20.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
<tr>
<td>InnerException</td>
<td>Inherited from System.SystemException.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 System.SystemException.Exception</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Inherited from System.SystemException.Exception</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleTruncateException Class
- OracleTruncateException Members
OracleTruncateException Methods

The `OracleTruncateException` methods are listed in Table 15–21.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from <code>System.Object</code> (Overloaded)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>Inherited from <code>System.SystemException.Exception</code></td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from <code>System.Object</code></td>
</tr>
<tr>
<td>GetObjectData</td>
<td>Inherited from <code>System.SystemException.Exception</code></td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from <code>System.Object</code></td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from <code>OracleTypeException</code></td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleTruncateException Class`
- `OracleTruncateException Members`
This chapter describes the object-related classes and interfaces in the Oracle Data Provider for .NET that provide support for Oracle user-defined data types (UDT).

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

- OracleCustomTypeMappingAttribute Class
- OracleObjectMappingAttribute Class
- OracleArrayMappingAttribute Class
- IOracleCustomType Interface
- IOracleCustomTypeFactory Interface
- IOracleArrayTypeFactory Interface
- OracleUdt Class
- OracleRef Class
- OracleUdtFetchOption Enumeration
- OracleUdtStatus Enumeration
OracleCustomTypeMappingAttribute Class

The OracleCustomTypeMappingAttribute class is used to mark a custom type factory class or struct with information that is used by ODP.NET when a custom type is used to represent an Oracle UDT.

Class Inheritance
System.Object
    System.Attribute
        System.OracleCustomTypeMappingAttribute

Declaration
// C#
[AttributeUsageAttribute(AttributeTargets.Class|AttributeTargets.Struct,
    AllowMultiple=false, Inherited=true)]
public sealed class OracleCustomTypeMappingAttribute : Attribute

Remarks
The OracleCustomTypeMapping attribute must be specified on the custom type factory class to indicate the Oracle UDT that the corresponding custom type represents. The Oracle UDT may be specified in the form schema_name.type_name.

For each Oracle UDT that the application uses, there must be a unique custom type factory, as follows:

- **Oracle Object Types**
  The custom type factory must return a custom type that cannot be used to represent any other Oracle Object Type.

- **Oracle Collection Types**
  The custom type factory may return a custom type that can be used by other Oracle Collection Types. This is common when an array type is used to represent an Oracle Collection, that is, when an int[] is used to represent a collection of NUMBERS.

If the OracleCustomTypeMappingAttribute is not specified, then custom type mappings must be specified through an XML configuration file, for example, app.config for Windows applications or the web.config for web applications, and the machine.config

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleCustomTypeMappingAttribute Members
- OracleCustomTypeMappingAttribute Constructors
- OracleCustomTypeMappingAttribute Static Methods
- OracleCustomTypeMappingAttribute Methods
OracleCustomTypeMappingAttribute Members

OracleCustomTypeMappingAttribute members are listed in the following tables.

OracleCustomTypeMappingAttribute Constructors
OracleCustomTypeMappingAttribute constructors are listed in Table 16–1.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleCustomTypeMappingAttribute Constructors</td>
<td>Instantiates a new instance of OracleCustomTypeMappingAttribute class</td>
</tr>
</tbody>
</table>

OracleCustomTypeMappingAttribute Static Methods
OracleCustomTypeMappingAttribute static methods are listed in Table 16–2.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttributes</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefined</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

OracleCustomTypeMappingAttribute Properties
OracleCustomTypeMappingAttribute properties are listed in Table 16–3.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UdtTypeName</td>
<td>Specifies the Oracle user-defined type name that the custom class maps to</td>
</tr>
<tr>
<td>TypeId</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

OracleCustomTypeMappingAttribute Methods
OracleCustomTypeMappingAttribute methods are listed in Table 16–4.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Match</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleCustomTypeMappingAttribute Class
OracleCustomTypeMappingAttribute Constructors

OracleCustomTypeMappingAttribute constructors create new instances of the OracleCustomTypeMappingAttribute class.

Overload List:

- **OracleCustomTypeMappingAttribute(string)**
  
  This constructor creates and initializes an OracleCustomTypeMappingAttribute using the specified Oracle user-defined type name.

  **See Also:**
  - "Oracle.DataAccess.Types Namespace" on page 1-9
  - OracleCustomTypeMappingAttribute Class
  - OracleCustomTypeMappingAttribute Methods

OracleCustomTypeMappingAttribute(string)

This constructor creates and initializes an OracleCustomTypeMappingAttribute using the specified Oracle user-defined type name.

**Declaration**

```csharp
// C#
public OracleCustomTypeMappingAttribute(string udtTypeName)
```

**Parameters**

- **udtTypeName**
  
  The Oracle user-defined type name that the custom class maps to.

**Remarks**

The **udtTypeName** parameter is case-sensitive. The **udtTypeName** is specified in the form of `schema_name.type_name`.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleCustomTypeMappingAttribute Class
- OracleCustomTypeMappingAttribute Members
OracleCustomTypeMappingAttribute Static Methods

OracleCustomTypeMappingAttribute static methods are listed in Table 16–5.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttributes</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefined</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleCustomTypeMappingAttribute Class
- OracleCustomTypeMappingAttribute Members
OracleCustomTypeMappingAttribute Properties

OracleCustomTypeMappingAttribute properties are listed in Table 16–6.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UdtTypeName</td>
<td>Specifies the Oracle user-defined type name that the custom class maps to</td>
</tr>
<tr>
<td>TypeId</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleCustomTypeMappingAttribute Class
- OracleCustomTypeMappingAttribute Members

UdtTypeName

This property specifies the Oracle user-defined type name that the custom class maps to.

Declaration

```csharp
public string UdtTypeName {get; set;}
```

Property Value

A string that represents an Oracle user-defined type name.

Remarks

UdtTypeName is case-sensitive. It is specified in the form of `schema_name.type_name`.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleCustomTypeMappingAttribute Class
- OracleCustomTypeMappingAttribute Members
OracleCustomTypeMappingAttribute Methods

OracleCustomTypeMappingAttribute methods are listed in Table 16–7.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Match</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleCustomTypeMappingAttribute Class
- OracleCustomTypeMappingAttribute Members
OracleObjectMappingAttribute Class

The OracleObjectMappingAttribute class marks custom class fields or properties with information that ODP.NET uses when a custom type represents an Oracle Object type.

Class Inheritance
System.Object
    System.Attribute
        System.OracleObjectMappingAttribute

Declaration
// C#
[AttributeUsageAttribute(AttributeTargets.Field|AttributeTargets.Property,
    AllowMultiple=false, Inherited=true)]

public sealed class OracleObjectMappingAttribute : Attribute

Remarks
The OracleObjectMappingAttribute is specified on members of a custom type that represent an Oracle object type. This attribute must specify the name or zero-based index of the attribute in the Oracle object that the custom class field or property maps to. This also allows the custom type to declare field or property names which differ from the Oracle Object type.

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Members
- OracleObjectMappingAttribute Constructors
- OracleObjectMappingAttribute Static Methods
- OracleObjectMappingAttribute Properties
- OracleObjectMappingAttribute Methods
OracleObjectMappingAttribute Members

OracleObjectMappingAttribute members are listed in the following tables.

OracleObjectMappingAttribute Constructors
OracleObjectMappingAttribute constructors are listed in Table 16–8.

Table 16–8  OracleObjectMappingAttribute Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleObjectMappingAttribute</td>
<td>Instantiates a new instance of OracleObjectMappingAttribute class (Overloaded)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
OracleObjectMappingAttribute Constructors

OracleObjectMappingAttribute constructors create new instances of the OracleObjectMappingAttribute class.

Overload List:

- **OracleObjectMappingAttribute(string)**
  This constructor creates and initializes an OracleObjectMappingAttribute object with the specified Oracle Object attribute name.

- **OracleObjectMappingAttribute(int)**
  This constructor creates and initializes an OracleObjectMappingAttribute with the specified Oracle Object attribute index.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
- OracleObjectMappingAttribute Members

**OracleObjectMappingAttribute(string)**

This constructor creates and initializes an OracleObjectMappingAttribute object with the specified Oracle Object attribute name.

**Declaration**

```
// C#
public OracleObjectMappingAttribute(string attrName);
```

**Parameters**

- **attrName**

  The name of the Oracle Object attribute to map to.

**Remarks**

The `attrName` parameter is case-sensitive.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
- OracleObjectMappingAttribute Members

**OracleObjectMappingAttribute(int)**

This constructor creates and initializes an OracleObjectMappingAttribute object with the specified Oracle Object attribute index.

**Declaration**

```
// C#
public OracleObjectMappingAttribute(int attrIndex);
```
OracleObjectMappingAttribute Constructors

Parameters

- attrIndex

  The zero-based index of the Oracle Object attribute to map to.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
- OracleObjectMappingAttribute Members
OracleObjectMappingAttribute Static Methods

OracleObjectMappingAttribute static methods are listed in Table 16–12.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttributes</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefined</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
- OracleObjectMappingAttribute Members
OracleObjectMappingAttribute Properties

OracleObjectMappingAttribute properties are listed in Table 16–13.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeIndex</td>
<td>Specifies the index of the Oracle Object attribute that must be retrieved</td>
</tr>
<tr>
<td>AttributeName</td>
<td>Specifies the name of the Oracle Object attribute that must be retrieved</td>
</tr>
<tr>
<td>TypeId</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
- OracleObjectMappingAttribute Members

AttributeIndex

This property specifies the index of the Oracle Object attribute that must be retrieved.

Declaration

```csharp
// C#
public int AttributeIndex {get;}
```

Property Value

The zero-based index of an Oracle Object type attribute.

Remarks

The AttributeIndex property specifies the index of the Oracle Object type attribute that the custom class field or property maps to. This allows the custom class to declare fields or property names that differ from the Oracle object.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
- OracleObjectMappingAttribute Members

AttributeName

This property specifies the name of the Oracle Object attribute that must be retrieved.

Declaration

```csharp
// C#
public string AttributeName {get;}
```
Property Value
The name of an attribute of an Oracle Object type.

Remarks
The AttributeName property specifies name of the attribute in the Oracle Object type that the custom class field or property maps to. This allows the custom class to declare field or property names that differ from the Oracle object.

The specified attribute name is case-sensitive.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
- OracleObjectMappingAttribute Members
OracleObjectMappingAttribute Methods

OracleObjectMappingAttribute methods are listed in Table 16–14.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Match</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleObjectMappingAttribute Class
- OracleObjectMappingAttribute Members
OracleArrayMappingAttribute Class

The OracleArrayMappingAttribute class is required to mark a custom class field or property with information that ODP.NET uses when a custom type represents an Oracle Collection type.

Class Inheritance
System.Object
    System.Attribute
        System.OracleArrayMappingAttribute

Declaration
[AttributeUsageAttribute(AttributeTargets.Field | AttributeTargets.Property, AllowMultiple=false, Inherited=true)]

// C#
public sealed class OracleArrayMappingAttribute : Attribute

Remarks
An OracleArrayMappingAttribute object must be specified when a custom type represents an Oracle Collection. This attribute is applied only to the custom class member that stores the collection elements.

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleArrayMappingAttribute Members
- OracleArrayMappingAttribute Constructors
- OracleArrayMappingAttribute Static Methods
- OracleArrayMappingAttribute Properties
- OracleArrayMappingAttribute Methods
OracleArrayMappingAttribute Members

OracleArrayMappingAttribute members are listed in the following tables.

OracleArrayMappingAttribute Constructors
OracleArrayMappingAttribute constructors are listed in Table 16–15.

Table 16–15  OracleArrayMappingAttribute Constructors

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

OracleArrayMappingAttribute Static Methods
OracleArrayMappingAttribute static methods are listed in Table 16–16.

Table 16–16  OracleArrayMappingAttribute Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttributes</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefined</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

OracleArrayMappingAttribute Properties
OracleArrayMappingAttribute properties are listed in Table 16–17.

Table 16–17  OracleArrayMappingAttribute Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeId</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

OracleArrayMappingAttribute Methods
OracleArrayMappingAttribute methods are listed in Table 16–18.

Table 16–18  OracleArrayMappingAttribute Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Match</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>
See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleArrayMappingAttribute Class
OracleArrayMappingAttribute Constructors

OracleArrayMappingAttribute constructors create new instances of the OracleArrayMappingAttribute class.

Overload List:

- OracleArrayMappingAttribute()
  This constructor creates and initializes an OracleArrayMappingAttribute object.

  See Also:
  - "Oracle.DataAccess.Types Namespace" on page 1-9
  - OracleArrayMappingAttribute Class
  - OracleArrayMappingAttribute Members

OracleArrayMappingAttribute()

This constructor creates and initializes an OracleArrayMappingAttribute object.

Declaration

// C#
public OracleArrayMappingAttribute();

Remarks

An OracleArrayMappingAttribute object must be applied when a custom class represents an Oracle Collection type, to specify the custom class field or property that stores the collection elements.

The OracleArrayMappingAttribute can be applied to only one field or property in the custom class.

  See Also:
  - "Oracle.DataAccess.Types Namespace" on page 1-9
  - OracleArrayMappingAttribute Class
  - OracleArrayMappingAttribute Members
OracleArrayMappingAttribute Static Methods

OracleArrayMappingAttribute static methods are listed in Table 16–19.

Table 16–19  OracleArrayMappingAttribute Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetCustomAttributes</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefined</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ReferenceEquals</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace” on page 1-9
- OracleArrayMappingAttribute Class
- OracleArrayMappingAttribute Members
OracleArrayMappingAttribute Properties

OracleArrayMappingAttribute properties are listed in Table 16–20.

Table 16–20 OracleArrayMappingAttribute Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeId</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleArrayMappingAttribute Class
- OracleArrayMappingAttribute Members
OracleArrayMappingAttribute Methods

OracleArrayMappingAttribute methods are listed in Table 16–21.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>Match</td>
<td>Inherited from System.Attribute</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Attribute</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleArrayMappingAttribute Class
- OracleArrayMappingAttribute Members
IOracleCustomType Interface

IOracleCustomType is an interface for converting between a Custom Type and an Oracle Object or Collection Type.

Declaration
// C#
public interface IOracleCustomType

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
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleCustomType Members
- IOracleCustomType Interface Methods
IOracleCustomType Members

IOracleCustomType members are listed in the following tables.

IOracleCustomType Interface Methods
IOracleCustomType interface methods are listed in Table 16–22.

Table 16–22 IOracleCustomType Interface Methods

<table>
<thead>
<tr>
<th>Interface Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FromCustomObject</td>
<td>Returns the values that set the Oracle Object attributes</td>
</tr>
<tr>
<td>ToCustomObject</td>
<td>Provides the Oracle Object with the attribute values to set on the custom type</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleCustomType Interface
IOracleCustomType Interface Methods

IOracleCustomType Interface methods are listed in Table 16–23.

Table 16–23  IOracleCustomType Interface Methods

<table>
<thead>
<tr>
<th>Interface Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FromCustomObject</td>
<td>Returns the values that set the Oracle Object attributes</td>
</tr>
<tr>
<td>ToCustomObject</td>
<td>Provides the Oracle Object with the attribute values to set on the custom type</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleCustomType Interface
- IOracleCustomType Members

FromCustomObject

This interface method creates an Oracle Object or Collection by setting the attribute or element values respectively on the specified Oracle UDT.

Declaration

// C#
void FromCustomObject(OracleConnection con, IntPtr pUdt);

Parameters

- con
  An OracleConnection instance.
- pUdt
  An opaque pointer to the Oracle Object or Collection to be created.

Remarks

The FromCustomObject method is used to build an Oracle Object or Collection from a custom object by setting attribute or element values respectively through the OracleUdt.SetValue method.

The OracleUdt.SetValue method is invoked as follows:

- Oracle Object Type
  For a custom type that represents an Oracle Object Type, the OracleUdt.SetValue method must be invoked for each non-NULL attribute value that needs to be set.

- Oracle Collection Type
  For a custom type that represents an Oracle Collection Type, a single call to OracleUdt.SetValue method specifies the collection element values.
ToCustomObject

This interface initializes a custom object using the specified Oracle UDT.

Declaration

```csharp
// C#
void ToCustomObject (OracleConnection con, IntPtr pUdt);
```

Parameters

- `con`  
  An `OracleConnection` instance.
- `pUdt`  
  An opaque pointer to the Oracle UDT.

Remarks

The `ToCustomObject` method is used to initialize a custom object from the specified Oracle Object or Collection by retrieving attribute or element values respectively through the `OracleUdt.GetValue` method.

The `OracleUdt.GetValue` method is invoked as follows:

- **Oracle Object Type**
  For a custom type that represents an Oracle Object Type, the `OracleUdt.GetValue` method must be invoked for each attribute value to be retrieved.

- **For a custom type that represents an Oracle Collection Type**, a single call to `OracleUdt.GetValue` method retrieves the collection element values.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `IOracleCustomType Interface`
- `IOracleCustomType Members`
The **IOracleCustomTypeFactory** interface is used by ODP.NET to create custom objects that represent Oracle Objects or Collections.

**Declaration**

// C#
public interface IOracleCustomTypeFactory

**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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleCustomTypeFactory Members
- IOracleCustomTypeFactory Interface Methods
IOracleCustomTypeFactory Members

IOracleCustomTypeFactory members are listed in the following tables.

IOracleCustomTypeFactory Interface Methods
IOracleCustomTypeFactory interface methods are listed in Table 16–24.

Table 16–24  IOracleCustomTypeFactory Interface Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObject</td>
<td>Returns a new custom object to represent an Oracle Object or Collection</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleCustomTypeFactory Interface
IOracleCustomTypeFactory Interface Methods

IOracleCustomTypeFactory Interface methods are listed in Table 16–25.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObject</td>
<td>Returns a new custom object to represent an Oracle Object or Collection</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleCustomTypeFactory Interface
- IOracleCustomTypeFactory Members

CreateObject

This interface method returns a new custom object to represent an Oracle Object or Collection.

Declaration

```
// C#
IOracleCustomType CreateObject();
```

Return Value

An IOracleCustomType object.

Remarks

The CreateObject method is used to create a new instance of a custom object to represent an Oracle Object or Collection.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleCustomTypeFactory Interface
- IOracleCustomTypeFactory Members
The `IOracleArrayTypeFactory` interface is used by ODP.NET to create arrays that represent Oracle Collections.

**Declaration**

```
// C#
public interface IOracleArrayTypeFactory
```

**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`
- **ODP.NET Version:** ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**
- ["Oracle.DataAccess.Types Namespace" on page 1-9](#)
- `IOracleArrayTypeFactory Members`
- `IOracleArrayTypeFactory Interface Methods`
IOracleArrayTypeFactory Members

IOracleArrayTypeFactory members are listed in the following tables.

IOracleArrayTypeFactory Interface Methods
IOracleArrayTypeFactory interface methods are listed in Table 16–26.

Table 16–26  IOracleArrayTypeFactory Interface Methods

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateArray</td>
<td>Returns a new array of the specified length to store Oracle Collection elements</td>
</tr>
<tr>
<td>CreateStatusArray</td>
<td>Returns a newly allocated OracleUdtStatus array of the specified length that will be used to store the null status of the collection elements</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleArrayTypeFactory Interface
IOracleArrayTypeFactory Interface Methods

IOracleArrayTypeFactory Interface methods are listed in Table 16–27.

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateArray</td>
<td>Returns a new array of the specified length to store Oracle Collection elements</td>
</tr>
<tr>
<td>CreateStatusArray</td>
<td>Returns a newly allocated OracleUdtStatus array of the specified length that will be used to store the null status of the collection elements</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleArrayTypeFactory Interface
- IOracleArrayTypeFactory Members

CreateArray

This interface method returns a new array of the specified length to store Oracle Collection elements.

Declaration

```
// C#
Array CreateArray(int numElems);
```

Parameters

- `numElems`
  - The number of collection elements to be returned.

Return Value

A System.Array object.

Remarks

An Oracle Collection Type may be represented in either of the following ways:
- As an array of the appropriate type. The type must be able to represent a collection element.
- As a Custom Type that contains an array of the appropriate type.

In both cases, the `CreateArray` method creates an array of the specified length to store the collection elements.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleArrayTypeFactory Interface
- IOracleArrayTypeFactory Members
CreateStatusArray

This method returns a newly allocated OracleUdtStatus array of the specified length that will be used to store the null status of the collection elements.

Declaration

// C#
Array CreateStatusArray(int numElems);

Parameters

- numElems

  The number of collection elements to be returned.

Return Value

A multi-dimensional OracleUdtStatus array as a System.Array.

Remarks

An Oracle Collection Type can be represented in the following ways:

- As an array of the appropriate type. The type must be able to represent a collection element.
- As a Custom Type that contains an array of the appropriate type.

In both cases, the CreateStatusArray method creates an OracleUdtStatus array of the specified length that stores the null status of the collection elements.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- IOracleArrayTypeFactory Interface
- IOracleArrayTypeFactory Members
- "OracleUdtFetchOption Enumeration" on page 16-73
OracleUdt Class

The OracleUdt class defines static methods that are used when converting between Custom Types and Oracle UDTs and vice-versa.

**Class Inheritance**

System.Object
  System.OracleUdt

**Declaration**

public sealed class OracleUdt

**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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Members
- OracleUDT Static Methods
OracleUdt Members

OracleUdt static methods are listed in Table 16–28.

Table 16–28  OracleUdt Static Methods

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetValue</td>
<td>Gets the attributes or elements from the specified Oracle UDT (Overloaded)</td>
</tr>
<tr>
<td>IsDBNull</td>
<td>Indicates whether or not the specified attribute being retrieved is NULL (Overloaded)</td>
</tr>
<tr>
<td>SetValue</td>
<td>Sets the attributes or elements on the specified Oracle UDT (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
OracleUDT Static Methods

OracleUDT methods are listed in Table 16–29.

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetValue</td>
<td>Gets the attributes or elements from the specified Oracle UDT (Overloaded)</td>
</tr>
<tr>
<td>IsDBNull</td>
<td>Indicates whether or not the specified attribute being retrieved is NULL (Overloaded)</td>
</tr>
<tr>
<td>SetValue</td>
<td>Sets the attributes or elements on the specified Oracle UDT (Overloaded)</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- OracleUdt Members

GetValue

GetValue methods get the attributes or elements from the specified Oracle UDT.

Overload List:
- GetValue(OracleConnection, IntPtr, string)
  This method gets the attributes or elements from the specified Oracle UDT, using the specified attribute name.
- GetValue(OracleConnection, IntPtr, int)
  This method gets the attribute or elements from the specified Oracle UDT, using the specified index.
- GetValue(OracleConnection, IntPtr, string, out object)
  This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.
- GetValue(OracleConnection, IntPtr, int, out object)
  This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- "OracleUdt Members"

GetValue(OracleConnection, IntPtr, string)

This method gets the attributes or elements from the specified Oracle UDT, using the specified attribute name.
Declaration

public static object GetValue(OracleConnection con, IntPtr pUdt, string attrName);

Parameters

■ **con**
  An OracleConnection instance.

■ **pUdt**
  A pointer to an Oracle UDT.

■ **attrName**
  The case-sensitive name of the attribute to be retrieved. Null is specified for retrieving collection elements from a Custom Type that represents an Oracle Collection.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

ArgumentException - The specified name is not a valid attribute name.

Remarks

The IOracleCustomType.ToCustomObject method invokes OracleUdt.GetValue method passing it the **con** and **pUdt** parameters. The OracleUdt.GetValue method returns these types of object:

■ **Oracle Object Type**

  For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the OracleObjectMappingAttribute object.

■ **Oracle Collection Type**

  For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the OracleArrayMappingAttribute object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static Null property of the type is returned. For attributes that are represented as Nullable types, for example, System.String and System.Array Types, null is returned, and for all other remaining built-in types such as Int32 and DateTime DBNull.Value is returned.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9

■ OracleUdt Class

■ OracleUdt Members

■ "OracleUdtFetchOption Enumeration" on page 16-73
**GetValue(OracleConnection, IntPtr, int)**

This method gets the attribute or elements from the specified Oracle UDT, using the specified index.

**Declaration**

```csharp
// C#
public static object GetValue(OracleConnection con, IntPtr pUdt, int attrIndex);
```

**Parameters**

- **con**
  
  An OracleConnection instance.

- **pUdt**
  
  A pointer to an Oracle UDT.

- **attrIndex**
  
  The zero-based index of the attribute to be retrieved. For retrieving collection elements from a Custom Type that represents an Oracle Collection, zero must be specified.

**Return Value**

An object representing the returned attribute or collection elements.

**Exceptions**

ArgumentNullException - The specified index is not a valid attribute index.

**Remarks**

The IOracleCustomType.ToCustomObject method invokes OracleUdt.GetValue method passing it the `con` and `pUdt` parameters. The OracleUdt.GetValue method returns these types of object:

- **Oracle Object Type**
  
  For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the OracleObjectMappingAttribute object.

- **Oracle Collection Type**
  
  For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the OracleArrayMappingAttribute object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static Null property of the type is returned. For attributes that are represented as Nullable types, for example, `System.String` and `System.Array` Types, null is returned, and for all other remaining built-in types such as `Int32` and `DateTime`, `DBNull.Value` is returned.
GetValue(OracleConnection, IntPtr, string, out object)

This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

Declaration

// C#
public static object GetValue(OracleConnection con, IntPtr pUdt, string attrName,
   out object statusArray);

Parameters

- **con**
  An OracleConnection instance.

- **pUdt**
  An opaque pointer to an Oracle UDT.

- **attrName**
  The case-sensitive name of the attribute to be retrieved. Null must specified for retrieving collection elements from a Custom Type that represents an Oracle Collection.

- **statusArray** - The OracleUdtStatus array which returns the null status for the retrieved collection elements.

Return Value

An object representing the returned attribute or collection elements.

Exceptions

ArgumentException - The specified name is not a valid attribute name.

Remarks

The IOracleCustomType.ToCustomObject method invokes OracleUdt.GetValue method passing it the con and pUdt parameters. The OracleUdt.GetValue method returns these types of object:

- **Oracle Object Type**
  For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the OracleObjectMappingAttribute object.

- **Oracle Collection Type**

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- OracleUdt Members
- "OracleUdtFetchOption Enumeration" on page 16-73
For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the OracleArrayMappingAttribute object is applied.

In the case of NULL attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static Null property of the type is returned. For attributes that are represented as Nullable types, for example, System.String and System.Array Types, null is returned, and for all other remaining built-in types such as Int32 and DateTime DBNull.Value is returned.

If the collection being returned is not NULL, the output statusArray parameter is populated with the null status for each of the collection elements.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- OracleUdt Members
- "OracleUdtFetchOption Enumeration" on page 16-73

GetValue(OracleConnection, IntPtr, int, out object)

This method returns either the elements of the specified collection attribute of the specified Oracle Object or the elements of the specified Oracle Collection.

Declaration

// C#
public static object GetValue(OracleConnection con, IntPtr pUdt, int attrIndex, out object statusArray);

Parameters
- con
  An OracleConnection instance.
- pUdt
  An opaque pointer to an Oracle UDT.
- attrIndex
  The zero-based index of the attribute to be retrieved. For retrieving collection elements from a Custom Type that represents an Oracle Collection, 0 is specified.
- statusArray
  The OracleUdtStatus array which returns the null status for the retrieved collection elements.

Return Value
An object representing the returned attribute or collection elements.

Exceptions
ArgumentOutOfRangeException - The specified index is not a valid attribute index.
Remarks

The `IOracleCustomType.ToCustomObject` method invokes `OracleUdt.GetValue` method passing it the `con` and `pUdt` parameters. The `OracleUdt.GetValue` method returns these types of object:

- **Oracle Object Type**
  
  For a Custom Type that represents an Oracle Object Type, the type returned for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- **Oracle Collection Type**
  
  For a Custom Type that represents an Oracle Collection Type, the type returned is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

In the case of `NULL` attribute values, the appropriate null representation of the type is returned. For example, for attributes that are represented as Custom Types and Provider Specific Types, the static `null` property of the type is returned. For attributes that are represented as Nullable types, for example, `System.String` and `System.Array` Types, `null` is returned, and for all other remaining built-in types such as `Int32` and `DateTime`, `DBNull.Value` is returned.

If the collection being returned is not `NULL`, the output `statusArray` parameter is populated with the null status for each of the collection elements.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- OracleUdt Members
- "OracleUdtFetchOption Enumeration" on page 16-73

IsDBNull

`IsDBNull` methods indicate whether or not the specified attribute being retrieved is `NULL`.

Overload List:

- **IsDBNull(OracleConnection, IntPtr, string)**
  
  This method indicates whether or not the attribute being retrieved, specified by `OracleConnection`, pointer, and attribute name, is `NULL`.

- **IsDBNull(OracleConnection, IntPtr, int)**
  
  This method indicates whether or not the attribute being retrieved, specified by `OracleConnection`, pointer, and attribute index, is `NULL`.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- OracleUdt Members
- "OracleUdtFetchOption Enumeration" on page 16-73
IsDBNull(OracleConnection, IntPtr, string)

This method indicates whether or not the attribute being retrieved, specified by OracleConnection, pointer, and attribute name, is NULL.

Declaration

// C#
public static bool IsDBNull(OracleConnection con, IntPtr pUdt, string attrName);

Parameters

- **con**
  An OracleConnection instance.
- **pUdt**
  A pointer to an Oracle UDT.
- **attrName**
  The case-sensitive name of the attribute.

Return Value

True if the specified attribute is NULL; otherwise, false.

Exceptions

ArgumentException - The specified name is not a valid attribute name.

Remarks

This method is invoked from the IOracleCustomType.ToCustomObject method. The con and pUdt parameter is passed from the IOracleCustomType.ToCustomObject method to the OracleUdt.IsDBNull method. The attrName parameter is case-sensitive.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- OracleUdt Members

IsDBNull(OracleConnection, IntPtr, int)

This method indicates whether or not the attribute being retrieved, specified by OracleConnection, pointer, and attribute index, is NULL.

Declaration

// C#
public static bool IsDBNull(OracleConnection con, IntPtr pUdt, int attrIndex);

Parameters

- **con**
  An OracleConnection instance.
- **pUdt**

Oracle Data Provider for .NET UDT-Related Classes  16-45
An opaque pointer to an Oracle UDT.

- attrIndex
  The zero-based index of the attribute.

**Return Value**

True if the specified attribute is NULL; otherwise, false.

**Exceptions**

ArgumentNullException - The specified index is not a valid attribute index.

**Remarks**

This method is invoked from the IOracleCustomType.ToCustomObject method. The `con` and `pUdt` parameter is passed from the IOracleCustomType.ToCustomObject method to the OracleUdt.IsDBNull method.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- OracleUdt Members

**SetValue**

SetValue methods set the attributes or elements on the specified Oracle UDT.

**Overload List:**

- `SetValue(OracleConnection, IntPtr, string, object)`
  This method sets the attribute or elements on the specified Oracle UDT, using the specified attribute name and value.

- `SetValue(OracleConnection, IntPtr, int, object)`
  This method sets the attribute or elements on the specified Oracle UDT, using the specified index and value.

- `SetValue(OracleConnection, IntPtr, string, object, object)`
  This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

- `SetValue(OracleConnection, IntPtr, int, object, object)`
  This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- OracleUdt Members
SetValue(OracleConnection, IntPtr, string, object)

This method sets the attribute or elements on the specified Oracle UDT, using the specified attribute name and value.

Declaration

// C#
public static void SetValue(OracleConnection con, IntPtr pUdt, string attrName, object value);

Parameters

■ con
   An OracleConnection instance.
■ pUdt
   An opaque pointer to an Oracle UDT.
■ attrName
   The name of the attribute to be set. Specify null for setting collection elements from a Custom Type that represents an Oracle Collection.
■ value
   The attribute or collection value to be set.

Exceptions

ArgumentException - The specified value is not of the appropriate type.

Remarks

The IOracleCustomType.FromCustomObject method invokes OracleUdt.SetValue method passing it the con and pUdt parameters. The OracleUdt.SetValue method returns these types of object:

■ Oracle Object Type
   For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the OracleObjectMappingAttribute object.

■ Oracle Collection Type
   For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the OracleArrayMappingAttribute object is applied.

See Also:

■ "Oracle.DataAccess.Types Namespace" on page 1-9
■ OracleUdt Class
■ "OracleUdt Members"

SetValue(OracleConnection, IntPtr, int, object)

This method sets the attribute or elements on the specified Oracle UDT, using the specified index and value.
OracleUDT Static Methods

Declaration

```c#
public static void SetValue(OracleConnection con, IntPtr pUdt, int attrIndex, object value);
```

Parameters

- **con**
  An `OracleConnection` instance.

- **pUdt**
  An opaque pointer to an Oracle UDT.

- **attrIndex**
  The index of the attribute to be set. Specify 0 for setting collection elements from a Custom Type that represents an Oracle Collection.

- **value**
  The attribute or collection value to be set.

Exceptions

- `ArgumentException` - The specified value is not of the appropriate type.

Remarks

The `IOrganizationCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the `con` and `pUdt` parameters. The `OracleUdt.SetValue` method returns these types of object:

- **Oracle Object Type**
  For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- **Oracle Collection Type**
  For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleUdt Class`
- "OracleUdt Members"

**SetValue(OracleConnection, IntPtr, string, object, object)**

This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

Declaration

```c#
public static void SetValue(OracleConnection con, IntPtr pUdt, string attrName,
```

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleUdt Class`
- "OracleUdt Members"
object value, object statusArray);

Parameters

- **con**
  An OracleConnection instance.

- **pUdt**
  An opaque pointer to an Oracle UDT.

- **attrName**
  The name of the attribute to be set. Specify null for setting collection elements from a Custom Type that represents an Oracle Collection.

- **value**
  The attribute or collection value to be set.

- **statusArray**
  The null status for the collection elements.

Exceptions

ArgumentException - The specified value is not of the appropriate type.

Remarks

The IOracleCustomType.FromCustomObject method invokes OracleUdt.SetValue method passing it the con and pUdt parameters. The OracleUdt.SetValue method returns these types of object:

- Oracle Object Type
  For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute name is the type of the member in the custom class or struct that is mapped to the attribute using the OracleObjectMappingAttribute object.

- Oracle Collection Type
  For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the OracleArrayMappingAttribute object is applied.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- "OracleUdt Members"

`SetValue(OracleConnection, IntPtr, int, object, object)`

This method sets either the specified collection attribute of the specified Oracle Object or elements of the specified Oracle Collection, to the specified value using the supplied null status of the collection elements.

Declaration

```c#
public static void SetValue(OracleConnection con, IntPtr pUdt, int attrIndex,
```
object value, object statusArray);

Parameters

- **con**
  An `OracleConnection` instance.
- **pUdt**
  An opaque pointer to an Oracle UDT.
- **attrIndex**
  The index of the attribute to be set. Specify 0 for setting collection elements from a Custom Type that represents an Oracle Collection.
- **value**
  The attribute or collection value to be set.
- **statusArray**
  The null status for the collection elements.

Exceptions

- **ArgumentException** - The specified value is not of the appropriate type.

Remarks

The `IOracleCustomType.FromCustomObject` method invokes `OracleUdt.SetValue` method passing it the `con` and `pUdt` parameters. The `OracleUdt.SetValue` method returns these types of object:

- **Oracle Object Type**
  For a Custom Type that represents an Oracle Object Type, the type accepted for a specified attribute index is the type of the member in the custom class or struct that is mapped to the attribute using the `OracleObjectMappingAttribute` object.

- **Oracle Collection Type**
  For a Custom Type that represents an Oracle Collection Type, the type accepted is the type of the member in the custom class or struct to which the `OracleArrayMappingAttribute` object is applied.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleUdt Class
- "OracleUdt Members"
OracleRef Class

An OracleRef instance represents an Oracle REF, which references a persistent, standalone, referenceable object that resides in the database. The OracleRef object provides methods to insert, update, and delete the Oracle REF.

Class Inheritance
System.Object
    System.MarshalByRefObject
        Oracle.DataAccess.Types.OracleRef

Declaration
// C#
public sealed class OracleRef : MarshalByRefObject, ICloneable, IDisposable, INullable

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

Remarks
If two or more OracleRef objects that refer to the same Oracle object in the database are retrieved through the same OracleConnection, then their operations on the referenced object must be synchronized.

Requirements
Namespace: Oracle.DataAccess.Types
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Members
- OracleRef Constructors
- OracleRef Static Methods
- OracleRef Instance Properties
- OracleRef Instance Methods
OracleRef Members

OracleRef members are listed in the following tables.

OracleRef Constructors
OracleRef constructors are listed in Table 16–30.

**Table 16–30** OracleRef Constructors

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

OracleRef Static Fields
OracleRef static methods are listed in Table 16–31.

**Table 16–31** OracleRef Static Fields

<table>
<thead>
<tr>
<th>Static Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an OracleRef instance</td>
</tr>
</tbody>
</table>

OracleRef Static Methods
OracleRef static methods are listed in Table 16–32.

**Table 16–32** OracleRef Static Methods

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

OracleRef Instance Properties
OracleRef instance properties are listed in Table 16–33.

**Table 16–33** OracleRef Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>References the connection used by the OracleRef</td>
</tr>
<tr>
<td>HasChanges</td>
<td>References the connection used by the OracleRef</td>
</tr>
<tr>
<td>IsLocked</td>
<td>Indicates whether or not the REF is locked</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the Oracle REF is NULL</td>
</tr>
<tr>
<td>ObjectTableName</td>
<td>Returns the fully qualified object table name that is associated with the REF</td>
</tr>
<tr>
<td>Value</td>
<td>Returns a .NET representation of this Oracle REF</td>
</tr>
</tbody>
</table>

OracleRef Instance Methods
OracleRef instance methods are listed in Table 16–34.

**Table 16–34** OracleRef Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Clones the REF</td>
</tr>
</tbody>
</table>
**Table 16–34 (Cont.) OracleRef Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>Deletes the referenced object from the database</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated for the OracleRef instance</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Flush</td>
<td>Flushes changes made on the REF object to the database</td>
</tr>
<tr>
<td>GetCustomObject</td>
<td>Returns the object that the specified REF references as a custom type (Overloaded)</td>
</tr>
<tr>
<td>GetCustomObjectForUpdate</td>
<td>Returns the object that the specified REF references as a custom type (Overloaded)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares two OracleREF objects</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the REF in the database</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Update</td>
<td>Updates the object referenced by the specified REF in the database using the specified custom object</td>
</tr>
</tbody>
</table>

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
OracleRef Constructors

OracleRef constructors instantiate new instances of OracleRef class.

Overload List:

- OracleRef(OracleConnection, string)
  This constructor creates an instance of the OracleRef class with a connection and a HEX string that represents an REF instance in the database.

- OracleRef(OracleConnection, string, string)
  This constructor creates an instance of the OracleRef class using the specified OracleConnection object, user-defined type name, and an object table name.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

OracleRef(OracleConnection, string)

This constructor creates an instance of the OracleRef class with a connection and a HEX string that represents an REF instance in the database.

Declaration

// C#
public OracleRef(OracleConnection con, string hexStr);

Parameters

- con
  An OracleConnection instance.

- hexStr
  A HEX string that represents an REF instance in the database.

Exceptions

ArgumentException - The HEX string does not represent a valid REF in the database.

ArgumentNullException - The connection or HEX string is null.

InvalidOperationException - The OracleConnection object is not open.

Remarks

When an OracleRef instance is created, it is referenced to a specific table in the database.
The connection must be opened explicitly by the application. OracleRef does not open the connection implicitly.
OracleRef Class

OracleRef(OracleConnection, string, string)

This constructor creates an instance of the OracleRef class using the specified OracleConnection object, user-defined type name, and an object table name.

Declaration

// C#
public OracleRef(OracleConnection con, string udtTypeName, string objTabName);

Parameters

- **con**
  An OracleConnection instance.

- **udtTypeName**
  A user-defined type name.

- **objTabName**
  An object table name.

Exceptions

ArgumentException - The object type name or the object table name is not valid.

ArgumentNullException - The object type name or the table name is null.

InvalidOperationException - The OracleConnection object is not open.

Remarks

When an OracleRef instance is created, this OracleRef instance is associated with the specific table in the database. In other words, it represents a persistent REF.

This constructor creates a reference to the object table. However, it does not cause any entries to be made in database tables until the object is flushed to the database, that is, until the OracleRef.Flush or the OracleConnection.FlushCache method is called on the OracleRef Connection. Therefore, any operation that attempts to operate on the database copy of the object before flushing the object, such as, lock the object or fetch the latest copy of the object from the database, results in an OracleException.

The connection must be opened explicitly by the application. OracleRef does not open the connection implicitly.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

See Also:

- "FlushCache” on page 5-99
OracleRef Static Fields

OracleRef static fields are listed in Table 16–35.

Table 16–35  OracleRef Static Fields

<table>
<thead>
<tr>
<th>Static Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Represents a null value that can be assigned to an OracleRef instance</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

Null

This static field represents a null value that can be assigned to an OracleRef instance.

Declaration

```csharp
// C#
public static readonly OracleRef Null;
```

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
OracleRef Static Methods

OracleRef static methods are listed in Table 16–36.

Table 16–36  OracleRef Static Methods

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

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
OracleRef Instance Properties

OracleRef instance properties are listed in Table 16–37.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>References the connection used by the OracleRef</td>
</tr>
<tr>
<td>HasChanges</td>
<td>References the connection used by the OracleRef</td>
</tr>
<tr>
<td>IsLocked</td>
<td>Indicates whether or not the REF is locked</td>
</tr>
<tr>
<td>IsNull</td>
<td>Indicates whether or not the Oracle REF is NULL</td>
</tr>
<tr>
<td>ObjectTableName</td>
<td>Returns the fully qualified object table name that is associated with the REF</td>
</tr>
<tr>
<td>Value</td>
<td>Returns a .NET representation of this Oracle REF</td>
</tr>
</tbody>
</table>

Connection

This instance property references the connection used by the OracleRef.

Declaration

// C#
public OracleConnection Connection{get;}

Property Value

An OracleConnection object associated with the REF.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

Once the Dispose method is invoked, this property is set to null.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

HasChanges

This instance property indicates whether or not the object referenced by the Oracle REF in the object cache has any changes that can be flushed to the database.
Declaration

// C#
public bool HasChanges {get;}

Property Value

Returns true if the object referenced by the Oracle REF in the object cache has any changes that can be flushed to the database; otherwise, returns false.

Exceptions

ObjectDisposedException - The object is already disposed.

Remarks

This property returns true if a copy of the referenced object in the object cache is updated or deleted.

If there is no copy of the referenced object in the object cache, the latest copy of the referenced object in the database is cached in the object cache and false is returned.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

IsLocked

This instance property indicates whether or not the REF is locked.

Declaration

// C#
public bool IsLocked {get;}

Property Value

Returns true if the REF is locked; otherwise returns false.

Exceptions

ObjectDisposedException - The object is already disposed.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

IsNull

This instance property indicates whether or not the Oracle REF is NULL.

Declaration

// C#
public bool IsNull {get;}

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
Oracle Ref Instance Properties

**Property Value**

Returns true if the REF is NULL; otherwise, returns false.

**Exceptions**

ObjectDisposedException - The object is already disposed.

**Remarks**

If the Oracle REF is NULL, this property returns true. Otherwise, it returns false.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

**ObjectTableName**

This instance property returns the fully-qualified object table name that is associated with the REF.

**Declaration**

// C#
public string ObjectTableName{get;}

**Property Value**

A fully-qualified object table name that is associated with the REF.

**Exceptions**

ObjectDisposedException - The object is already disposed.

**Remarks**

The object table name is in the form schema_Name.Table_Name.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

**Value**

This instance property returns a .NET representation of this Oracle REF.

**Declaration**

// C#
public string Value{get;}

**Property Value**

A .NET representation of the Oracle REF.
Exceptions
ObjectDisposedException - The object is already disposed.

Remarks
This property returns a HEX string that represents the REF.
The returned string can be used to create a new OracleRef instance by using the
OracleRef(OracleConnection, string) constructor.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
- "OracleRef(OracleConnection, string)" on page 16-54
Oracle Ref Instance Methods

OracleRef instance methods are listed in Table 16–38.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Clones the REF</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the referenced object from the database</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases resources allocated for the OracleRef instance</td>
</tr>
<tr>
<td>Equals</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Flush</td>
<td>Flushes changes made on the REF object to the database</td>
</tr>
<tr>
<td>GetCustomObject</td>
<td>Returns the object that the specified REF references as a custom type</td>
</tr>
<tr>
<td>GetCustomObjectForUpdate</td>
<td>Returns the object that the specified REF references as a custom type</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>GetType</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares two OracleRef objects</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the REF in the database</td>
</tr>
<tr>
<td>ToString</td>
<td>Inherited from System.Object</td>
</tr>
<tr>
<td>Update</td>
<td>Updates the object referenced by the specified REF in the database using the specified custom object</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

Clone

This instance method clones the REF.

Declaration

```csharp
// C#
public OracleRef Clone();
```

Return Value

A clone of the current instance.

Implements

ICloneable

Exceptions

InvalidOperationException - The associated connection is not open.
Delete

This method deletes the referenced object from the database.

Declaration

```csharp
// C#
public void Delete(bool bFlush);
```

Parameters

- **bFlush**

  A `bool` that specifies whether or not the `REF` is flushed immediately.

Remarks

This method marks the specified `REF` for deletion. Depending on whether the value of `bFlush` is set to `true` or `false`, the following occurs:

- **True**

  The object referenced by the specified `REF` is deleted immediately from the database.

  Before flushing objects, it is required that the application has explicitly started a transaction by executing the `BeginTransaction` method on the `OracleConnection` object. This is because if the object being flushed has not already been locked by the application, an exclusive lock is obtained implicitly for the object. The lock is only released when the transaction commits or rollbacks.

- **False**

  The object referenced by the `REF` is not deleted immediately from the database, but only when a subsequent `Flush` method is invoked for the specified `REF` or the `FlushCache` method is invoked on the `OracleRef` or the `FlushCache` method is invoked on the `OracleRef` connection.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleRef Class`
- `OracleRef Members`

Dispose

This instance method releases resources allocated for the `OracleRef` instance.

Declaration

```csharp
// C#
```
public void Dispose();

### Implements

IDisposable

### Remarks

The object cannot be reused after it is disposed. Although some properties can still be accessed, their values may not be up-to-date.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

### Flush

This instance method flushes changes made on the REF object to the database, such as updates or deletes.

#### Declaration

// C#
public void Flush();

#### Exceptions

InvalidOperationException - The associated connection is not open.

#### Remarks

Before flushing objects, it is required that the application has explicitly started a transaction by executing the BeginTransaction method on the OracleConnection object. This is because if the object being flushed has not already been locked by the application, an exclusive lock is obtained implicitly for the object. The lock is only released when the transaction commits or rollbacks.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

### GetCustomObject

GetCustomObject methods return the object that the specified REF references as a custom type.

#### Overload List

- GetCustomObject(OracleUdtFetchOption)
  
  This method returns the object that the specified REF references as a custom type using the specified fetch option.
- GetCustomObject(OracleUdtFetchOption, int)
This method returns the object that the specified REF references as a custom type using the specified fetch option and depth level.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

### GetCustomObject(OracleUdtFetchOption)

This method returns the object that the specified REF references, as a custom type, using the specified fetch option.

**Declaration**

```csharp
// C#
public object GetCustomObject(OracleUdtFetchOption fetchOption);
```

**Parameters**

- `fetchOption`
  
  An OracleUdtFetchOption value.

**Return Value**

A custom object that represents the object that the specified REF references.

**Exceptions**

InvalidOperationException - The specified connection is not open, or a valid custom type has not been registered for the type of the referenced object.

**Remarks**

This method returns a custom type determined by the UDT mappings on the specified connection.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

The application can use the `OracleUdtFetchOption` method to control the copy of the object that is returned according to the specified option:

- **OracleUdtFetchOption.Cache option**
  
  If this option is specified, and there is a copy of the referenced object in the object cache, it is returned immediately. If no cached copy exists, the latest copy of the referenced object in the database is cached in the object cache and returned.

- **OracleUdtFetchOption.Server option**
  
  If this option is specified, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.

- **OracleUdtFetchOption.TransactionCache option**
  
  If this option is specified, and a copy of the referenced object is cached in the current transaction, the copy is returned. Otherwise, the latest copy of the referenced object from the database is cached in the object cache and returned. If a
copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.

Note that if a cached copy of the referenced object was modified before the current transaction began, that is, if the `OracleRef.HasChanges` property returns `true`, then the `Recent` option returns the cached copy of the referenced object. Outside of a transaction, the `Recent` option behaves like the `Any` option.

**See Also:**
- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleRef` Class
- `OracleRef Members`
- "OracleUdtFetchOption Enumeration" on page 16-73

### GetCustomObject(OracleUdtFetchOption, int)

This method returns the object that the specified `REF` references, as a custom type, using the specified fetch option and depth level.

**Declaration**

```csharp
// C#
public object GetCustomObject(OracleUdtFetchOption fetchOption, int depthLevel);
```

**Parameters**

- `fetchOption`  
  An `OracleUdtFetchOption` value.
- `depthLevel`  
  The number of levels to be fetched for nested `REF` attributes.

**Return Value**

A custom object that represents the object that the specified `REF` references.

**Exceptions**

`InvalidOperationException` - The specified connection is not open, or a valid custom type has not been registered for the type of the referenced object.

**Remarks**

This method returns a custom type determined by the UDT mappings on the specified connection.

If the object that the `REF` references contains nested `REF` attributes, the `depthLevel` can be specified to optimize the subsequent object retrieval. The value of `depthLevel` determines the number of levels that are optimized.

For example, if the `depthLevel` is specified as two, the optimization is applied to all top-level nested `REF` attributes in the object being fetched and also to all nested `REF` attributes within the objects referenced by the top-level nested `REF` attributes.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.
The application can use the `OracleUdtFetchOption` method to control the copy of the Object that is returned according to the specified option:

- **OracleUdtFetchOption.Cache option**
  
  If this option is specified, and there is a copy of the referenced object in the object cache, it is returned immediately. If no cached copy exists, the latest copy of the referenced object in the database is cached in the object cache and returned.

- **OracleUdtFetchOption.Server option**
  
  If this option is specified, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.

- **OracleUdtFetchOption.TransactionCache option**
  
  If this option is specified, and a copy of the referenced object is cached in the current transaction, the copy is returned. Otherwise, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.

Note that if a cached copy of the referenced object was modified before the current transaction began, that is, if the `OracleRef.HasChanges` property returns `true`, then the `Recent` option returns the cached copy of the referenced object. Outside of a transaction, the `Recent` option behaves like the `Any` option.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
- "OracleUdtFetchOption Enumeration" on page 16-73

**GetCustomObjectForUpdate**

`GetCustomObjectForUpdate` methods return the object that the specified REF references as a custom type.

- **GetCustomObjectForUpdate(bool)**
  
  This method locks the specified REF in the database and returns the object that the specified REF references as a custom type using the specified wait option.

- **GetCustomObjectForUpdate(bool, int)**
  
  This method locks the specified REF in the database and returns the object that the specified REF references as a custom type using the specified wait option and depth level.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
GetCustomObjectForUpdate(bool)

This method locks the specified REF in the database and returns the object that the specified REF references, as a custom type, using the specified wait option.

**Declaration**

```csharp
public object GetCustomObjectForUpdate(bool bWait);
```

**Parameters**

- **bWait**
  
  Specifies if the REF is to be locked with the no-wait option. If wait is set to true, this method invocation does not return until the REF is locked.

**Return Value**

A custom object that represents the object that the specified REF references.

**Exceptions**

- **InvalidOperationException** - The specified connection is not open, or a valid custom type has not been registered for type of the referenced object.
- **OracleException** - bWait is set to false, and the lock cannot be acquired.

**Remarks**

This method returns the latest copy of the referenced object, as a custom type, determined by the custom types registered on the OracleRef connection.

To be able to release the lock on the REF appropriately after flushing the REF using the Flush method on the OracleRef or FlushCache method on the OracleConnection, the application must commit or rollback the transaction.

Therefore, it is required that, before invoking this method, a transaction is explicitly started by executing the BeginTransaction method on the OracleConnection object.

This method makes a network round-trip to lock the REF in the database. After this call, programmers can modify the associated row object exclusively. Then a call to the Flush method on the OracleRef or FlushCache method on the OracleConnection flushes the changes to the database.

If true is passed, this method blocks until the lock can be acquired. If false is passed, this method immediately returns. If the lock cannot be acquired, an OracleException is thrown.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
GetCustomObjectForUpdate(bool, int)

This method locks the specified REF in the database and returns the object that the specified REF references, as a custom type, using the specified wait option and depth level.

Declaration

public object GetCustomObjectForUpdate(bool bWait, int depthlevel);

Parameters

- **bWait**
  A boolean value that specifies if the REF is to be locked with the no-wait option. If wait is set to true, this method invocation does not return until the REF is locked.

- **depthLevel**
  The number of levels to be fetched for nested REF attributes.

Return Value

A custom object that represents the object that the specified REF references.

Exceptions

- InvalidOperationException - The specified connection is not open, or a valid custom type has not been registered for type of the referenced object.
- OracleException - bWait is set to false, and the lock cannot be acquired.

Remarks

This method returns the latest copy of the referenced object, as a custom type, determined by the custom types registered on the OracleRef connection.

To be able to release the lock on the REF appropriately after flushing the REF using the Flush method on the OracleRef or FlushCache method on the OracleConnection, the application must commit or rollback the transaction. Therefore, it is required that, before invoking this method, a transaction is explicitly started by executing the BeginTransaction method on the OracleConnection object.

This method makes a network round-trip to lock the REF in the database. After this call, programmers can modify the associated row object exclusively. Then a call to the Flush method on the OracleRef or FlushCache method on the OracleConnection flushes the changes to the database.

If true is passed, this method blocks until the lock can be acquired. If false is passed, this method immediately returns. If the lock cannot be acquired, an OracleException is thrown.

If the object that the REF references contains nested REF attributes, the depthLevel can be specified to optimize the subsequent object retrieval. The value of depthLevel determines the number of levels that are optimized.

For example, if the depthLevel is specified as 2, the optimization is applied to all top-level nested REF attributes in the object being fetched and also to all nested REF attributes within the objects referenced by the top-level nested REF attributes.
Oracle Ref Instance Methods

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

IsEqual

This instance method compares two OracleRef objects.

Declaration

```
// C#
public bool IsEqual(OracleRef oraRef);
```

Parameters

- `oraRef`

  The provided OracleRef object.

Return Value

`bool`

Remarks

This instance method returns `true` if the OracleRef instance and the OracleRef parameter both reference the same object. Otherwise, it returns `false`.

See Also:
- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

Lock

This instance method locks the REF in the database.

Declaration

```
// C#
public bool Lock(bool bWait);
```

Parameters

- `bWait`

  Specifies if the lock is set to the no-wait option. If `bWait` is set to `true`, the method invocation does not return until the REF is locked.

Return Value

A boolean value that indicates whether or not the lock has been acquired.
Exceptions

InvalidOperationException - The associated connection is not open.
ObjectDisposedException - The object is already disposed.

Remarks

In order for the application to release the lock on the REF appropriately after the Flush invocation on the OracleRef or FlushCache methods, the application must commit or rollback the transaction. Therefore, it is required that, before invoking a lock on an OracleRef object, a transaction is explicitly started by executing the BeginTransaction method on the OracleConnection object.

This instance method makes a network round-trip to lock the REF in the database. After this call, programmers can modify the attribute values of the associated row object exclusively. Then a call to the Flush instance method on the OracleRef or FlushCache method on the OracleConnection flushes the changes to the database.

If true is passed, this method blocks, that is, does not return, until the lock is acquired. Consequently, the return value is always true.

If false is passed, this method immediately returns. The return value indicates true if the lock is acquired, and false if it is not.

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members

Update

This method updates the object referenced by the specified REF in the database using the specified custom object.

Declaration

// C#
public void Update(object customObject, bool bFlush);

Parameters

- customObject
  The custom object used to update the referenced object.

- bFlush
  A boolean that specifies if the changes must be flushed immediately. If bFlush is set to true, this method invocation flushes the changes immediately.

Exceptions

InvalidOperationException - The specified connection is not open or the custom object does not map to the type of referenced object.

Remarks

This method marks the specified REF for update. Depending on whether the value of bFlush is set to true or false, the following occurs:

- True
The object referenced by the specified `REF` is updated immediately in the database.

Before flushing objects, it is required that the application has explicitly started a transaction by executing the `BeginTransaction` method on the `OracleConnection` object. This is because if the object being flushed has not already been locked by the application, an exclusive lock is obtained implicitly for the object. The lock is only released when the transaction commits or rollbacks.

- **False**

The object referenced by the `REF` is not updated immediately in the database, but only when a subsequent `Flush` method is invoked for the specified `REF` or the `FlushCache` method is invoked for the specified connection.

The connection must be opened explicitly by the application. This method does not open the connection implicitly.

**See Also:**

- "Oracle.DataAccess.Types Namespace" on page 1-9
- `OracleRef Class`
- `OracleRef Members`
OracleUdtFetchOption Enumeration

OracleUdtFetchOption enumeration values specify how to retrieve a copy of the referenceable object.

Table 16–39 lists all the OracleUdtFetchOption enumeration values with a description of each enumerated value.

**Table 16–39  OracleUdtFetchOption Enumeration Values**

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache</td>
<td>If there is a copy of the referenced object in the object cache, it is returned immediately. If no cached copy exists, the latest copy of the referenced object in the database is cached in the object cache and returned.</td>
</tr>
<tr>
<td>Server</td>
<td>The latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one.</td>
</tr>
<tr>
<td>TransactionCache</td>
<td>If a copy of the referenced object is cached in the current transaction, the copy is returned. Otherwise, the latest copy of the referenced object from the database is cached in the object cache and returned. If a copy of the referenced object already exists in the cache, the latest copy overwrites the existing one. Note that if a cached copy of the referenced object was modified before the current transaction began, that is, if the OracleRef.HasChanges property returns true, then the Recent option returns the cached copy of the referenced object. Outside of a transaction, the Recent option behaves like the Any option.</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
OracleUdtStatus Enumeration

OracleUdtStatus enumeration values specify the status of an object attribute or collection element. An object attribute or a collection element can be a valid value or a null value.

Table 16-40 lists all the OracleUdtStatus enumeration values with a description of each enumerated value:

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Indicates that an object attribute or collection element is NULL.</td>
</tr>
<tr>
<td>NotNull</td>
<td>Indicates that a non-NULL value exists for the object attribute or collection element.</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Types Namespace" on page 1-9
- OracleRef Class
- OracleRef Members
This chapter describes Oracle Data Provider for .NET support for Bulk Copy operations.

---

**Note:** Oracle Data Provider for .NET bulk copy operations do not support loading of UDT type columns.

---

This chapter includes the following topics:

- [OracleBulkCopy Class](#)
- [OracleBulkCopyColumnMapping Class](#)
- [OracleBulkCopyColumnMappingCollection Class](#)
- [OracleBulkCopyOptions Enumeration](#)
- [OracleRowsCopiedEventHandler Delegate](#)
- [OracleRowsCopiedEventArgs Class](#)
OracleBulkCopy Class

An OracleBulkCopy object efficiently bulk loads or copies data into an Oracle table from another data source.

Class Inheritance
System.Object
    System.OracleBulkCopy

Declaration
// C#
public sealed class OracleBulkCopy : IDisposable

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

Remarks
The OracleBulkCopy class can be used to write data to Oracle database tables only. However, the data source is not limited to Oracle databases; any data source can be used, as long as the data can be loaded to a DataTable instance or read with an IDataReader instance.

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Members
- OracleBulkCopy Constructors
- OracleBulkCopy Properties
- OracleBulkCopy Public Methods
- OracleBulkCopy Events
OracleBulkCopy Members

OracleBulkCopy members are listed in the following tables.

**OracleBulkCopy Constructors**
OracleBulkCopy constructors are listed in Table 17–1.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleBulkCopy Constructors</td>
<td>OracleBulkCopy constructors create new instances of the OracleBulkCopy class</td>
</tr>
</tbody>
</table>

**OracleBulkCopy Properties**
OracleBulkCopy properties are listed in Table 17–2.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchSize</td>
<td>Specifies the number of rows to be sent as a batch to the database</td>
</tr>
<tr>
<td>BulkCopyOptions</td>
<td>Specifies the OracleBulkCopyOptions enumeration value that determines the behavior of the bulk copy operation</td>
</tr>
<tr>
<td>BulkCopyTimeout</td>
<td>Specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted</td>
</tr>
<tr>
<td>ColumnMappings</td>
<td>Specifies the column mappings between the data source and destination table</td>
</tr>
<tr>
<td>Connection</td>
<td>Specifies the OracleConnection object that the Oracle database uses to perform the bulk copy operation</td>
</tr>
<tr>
<td>DestinationTableName</td>
<td>Specifies the database table that the data is loaded in</td>
</tr>
<tr>
<td>NotifyAfter</td>
<td>Defines the number of rows to be processed before a notification event is generated</td>
</tr>
</tbody>
</table>

**OracleBulkCopy Public Methods**
OracleBulkCopy public methods are listed in Table 17–3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Closes the OracleBulkCopy instance</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases any resources or memory allocated by the object</td>
</tr>
<tr>
<td>WriteToServer</td>
<td>Copies rows to a destination table</td>
</tr>
</tbody>
</table>

**OracleBulkCopy Events**
OracleBulkCopy events are listed in Table 17–4.
Table 17–4  **OracleBulkCopy Events**

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleRowsCopied</td>
<td>Triggered every time the number of rows specified by the</td>
</tr>
<tr>
<td></td>
<td>OracleBulkCopy.NotifyAfter property has been processed</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
OracleBulkCopy Constructors

OracleBulkCopy constructors create new instances of the OracleBulkCopy class.

**Overload List:**

- **OracleBulkCopy(OracleConnection)**
  This constructor instantiates a new instance of OracleBulkCopy class using the specified connection and default value for OracleBulkCopyOptions.

- **OracleBulkCopy(string)**
  This constructor instantiates a new instance of OracleBulkCopy based on the supplied connectionString and default value for OracleBulkCopyOptions.

- **OracleBulkCopy(OracleConnection, OracleBulkCopyOptions)**
  This constructor instantiates a new instance of OracleBulkCopy using the specified connection object and OracleBulkCopyOptions value.

- **OracleBulkCopy(string, OracleBulkCopyOptions)**
  This constructor instantiates a new instance of OracleConnection based on the supplied connectionString and OracleBulkCopyOptions value.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members

**OracleBulkCopy(OracleConnection)**

This constructor instantiates a new instance of OracleBulkCopy class using the specified connection and default OracleBulkCopyOptions enumeration values.

**Declaration**

// C#
public OracleBulkCopy(OracleConnection connection);

**Parameters**

- connection
  The open instance of OracleConnection that performs the bulk copy operation.

**Exceptions**

- ArgumentException - The connection parameter is null.
- InvalidOperationException - The connection is not in the open state.

**Remarks**

The connection object passed to this constructor must be open. It remains open after the OracleBulkCopy instance is closed.

This constructor uses the default enumeration value OracleBulkCopyOptions.Default.
The Connection property is set to the supplied connection.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
- "OracleBulkCopyOptions Enumeration" on page 17-42

OracleBulkCopy(string)
This constructor instantiates a new instance of the OracleBulkCopy class by first creating an OracleConnection object based on the supplied connectionString, then initializing the new OracleBulkCopy object with the OracleConnection object and OracleBulkCopyOptions default value.

Declaration

```
// C#
public OracleBulkCopy(string connectionString);
```

Parameters
- connectionString

  The connection information used to connect to the Oracle database and perform the bulk copy operation.

Exception

ArgumentNullException - The connectionString parameter is null.
ArgumentException - The connectionString parameter is empty.

Remarks

The WriteToServer method opens the connection, if it is not already opened. The connection is automatically closed when the OracleBulkCopy instance is closed.

This constructor uses the default enumeration value OracleBulkCopyOptions.Default.

The Connection property is set to the OracleConnection object initialized using the supplied connectionString.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
- "OracleBulkCopyOptions Enumeration" on page 17-42

OracleBulkCopy(OracleConnection, OracleBulkCopyOptions)
This constructor instantiates a new instance of OracleBulkCopy using the specified connection object and OracleBulkCopyOptions value.
Declaration

// C#
public OracleBulkCopy(OracleConnection connection, OracleBulkCopyOptions copyOptions);

Parameters

- **connection**
  The open instance of an OracleConnection object that performs the bulk copy operation.

- **copyOptions**
  The combination of OracleBulkCopyOptions enumeration values that determine the behavior of the OracleBulkCopy object.

Exceptions

- ArgumentNullException - The connection parameter is null.
- InvalidOperationException - The connection is not in the open state.

Remarks

The connection passed to this constructor must be open. It remains open after the OracleBulkCopy instance is closed.

The Connection property is set to the supplied connection.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
- "OracleBulkCopyOptions Enumeration" on page 17-42

OracleBulkCopy(string, OracleBulkCopyOptions)

This constructor instantiates a new instance of the OracleBulkCopy class by first creating an OracleConnection object based on the supplied connectionString, then initializing the new OracleBulkCopy object with the OracleConnection object and the supplied OracleBulkCopyOptions enumeration values.

Declaration

// C#
public OracleBulkCopy(string connectionString, OracleBulkCopyOptions copyOptions);

Parameters

- **connectionString**
  The connection information used to connect to the Oracle database to perform the bulk copy operation.

- **copyOptions**
  The combination of OracleBulkCopyOptions enumeration values that determine the behavior of the bulk copy operation.
Exceptions

ArgumentNullException - The connectionString is null.

ArgumentException - The connectionString parameter is empty.

Remarks

The constructor uses the new instance of the OracleConnection class to initialize a new instance of the OracleBulkCopy class. The OracleBulkCopy instance behaves according to options supplied in the copyOptions parameter.

The connection is automatically closed when the OracleBulkCopy instance is closed.

The Connection property is set to an OracleConnection object initialized using the supplied connectionString.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
- "OracleBulkCopyOptions Enumeration" on page 17-42
OracleBulkCopy Properties

OracleBulkCopy properties are listed in Table 17–5.

Table 17–5  OracleBulkCopy Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchSize</td>
<td>Specifies the number of rows to be sent as a batch to the database</td>
</tr>
<tr>
<td>BulkCopyOptions</td>
<td>Specifies the OracleBulkCopyOptions enumeration value that determines the behavior of the bulk copy operation</td>
</tr>
<tr>
<td>BulkCopyTimeout</td>
<td>Specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted</td>
</tr>
<tr>
<td>ColumnMappings</td>
<td>Specifies the column mappings between the data source and destination table</td>
</tr>
<tr>
<td>Connection</td>
<td>Specifies the OracleConnection object that the Oracle database uses to perform the bulk copy operation</td>
</tr>
<tr>
<td>DestinationTableName</td>
<td>Specifies the database table that the data is loaded in</td>
</tr>
<tr>
<td>NotifyAfter</td>
<td>Defines the number of rows to be processed before a notification event is generated</td>
</tr>
</tbody>
</table>

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

**BatchSize**

This property specifies the number of rows to be sent as a batch to the database.

**Declaration**

```csharp
// C#
public int BatchSize {get; set;}
```

**Property Value**

An integer value for the number of rows to be sent to the database as a batch.

**Exceptions**

ArgumentOutOfRangeException - The batch size is less than zero.

**Remarks**

The default value is zero, indicating that the rows are not sent to the database in batches. The entire set of rows are sent in one single batch.

A batch is complete when BatchSize number of rows have been processed or there are no more rows to send to the database.

- If BatchSize > 0 and the UseInternalTransaction bulk copy option is specified, each batch of the bulk copy operation occurs within a transaction. If the
connection used to perform the bulk copy operation is already part of a transaction, an InvalidOperationException exception is raised.

- If BatchSize > 0 and the UseInternalTransaction option is not specified, rows are sent to the database in batches of size BatchSize, but no transaction-related action is taken.

The BatchSize property can be set at any time. If a bulk copy is already in progress, the current batch size is determined by the previous batch size. Subsequent batches use the new batch size.

If the BatchSize property is initially zero and changes while a WriteToServer operation is in progress, that operation loads the data as a single batch. Any subsequent WriteToServer operations on the same OracleBulkCopy instance use the new BatchSize.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
- "OracleBulkCopyOptions Enumeration" on page 17-42

**BulkCopyOptions**

This property specifies the OracleBulkCopyOptions enumeration value that determines the behavior of the bulk copy option.

**Declaration**

```csharp
// C#
public OracleBulkCopyOptions BulkCopyOptions {get; set;}
```

**Property Value**

The OracleBulkCopyOptions enumeration object that defines the behavior of the bulk copy operation.

**Exceptions**

ArgumentNullException - The bulk copy options set is null.

**Remarks**

The default value of this property is OracleBulkCopyOptions.Default value. This property can be used to change the bulk copy options between the batches of a bulk copy operation.

**See Also:**
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
- "OracleBulkCopyOptions Enumeration" on page 17-42
**BulkCopyTimeout**

This property specifies the number of seconds allowed for the bulk copy operation to complete before it is aborted.

**Declaration**

```csharp
// C#
public int BulkCopyTimeout {get; set;}
```

**Property Value**

An integer value for the number of seconds after which the bulk copy operation times out.

**Exceptions**

`ArgumentOutOfRangeException` - The timeout value is set to less than zero.

**Remarks**

The default value is 30 seconds.

If `BatchSize > 0`, rows that were sent to the database in the previous batches remain committed. The rows that are processed in the current batch are not sent to the database. If `BatchSize = 0`, no rows are sent to the database.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members

**ColumnMappings**

This property specifies the column mappings between the data source and destination table.

**Declaration**

```csharp
// C#
public OracleBulkCopyColumnMappingCollection ColumnMappings {get;}
```

**Property Value**

The `OracleBulkCopyColumnMappingCollection` object that defines the column mapping between the source and destination table.

**Remarks**

The `ColumnMappings` collection is unnecessary if the data source and the destination table have the same number of columns, and the ordinal position of each source column matches the ordinal position of the corresponding destination column. However, if the column counts differ, or the ordinal positions are not consistent, the `ColumnMappings` collection must be used to ensure that data is copied into the correct columns.

During the execution of a bulk copy operation, this collection can be accessed, but it cannot be changed.

By default, this property specifies an empty collection of column mappings.
OracleBulkCopy Properties

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

Connection

This property specifies the OracleConnection object that the Oracle database uses to perform the bulk copy operation.

Declaration

// C#
public OracleConnection Connection {get; }

Property Value

The OracleConnection object used for the bulk copy operations.

Remarks

This property gets the connection constructed by the OracleBulkCopy, if the OracleBulkCopy object is initialized using a connection string.

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

DestinationTableName

This property specifies the database table that the data is loaded into.

Declaration

// C#
public string DestinationTableName {get; set;}

Property Value

A string value that identifies the destination table name.

Exceptions

ArgumentNullException - The destination table name set is null.
ArgumentException - The destination table name is empty.

Remarks

If DestinationTableName is modified while a WriteToServer operation is running, the change does not affect the current operation. The new DestinationTableName value is used the next time a WriteToServer method is called.
**NotifyAfter**

This property defines the number of rows to be processed before a notification event is generated.

**Declaration**

```csharp
// C#
public int NotifyAfter {get; set;}
```

**Property Value**

An integer value that specifies the number of rows to be processed before the notification event is raised.

**Exceptions**

- `ArgumentOutOfRangeException` - The property value is set to a number less than zero.

**Remarks**

The default value for this property is zero, to specify that no notifications events are to be generated.

This property can be retrieved in user interface components to display the progress of a bulk copy operation. The `NotifyAfter` property can be set at anytime, even during a bulk copy operation. The changes take effect for the next notification and any subsequent operations on the same instance.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
OracleBulkCopy Public Methods

OracleBulkCopy methods are listed in Table 17–6.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Closes the OracleBulkCopy instance</td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases any resources or memory allocated by the object</td>
</tr>
<tr>
<td>WriteToServer</td>
<td>Copies rows to a destination table</td>
</tr>
</tbody>
</table>

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

Close

This method closes the OracleBulkCopy instance.

Declaration

```csharp
// C#
public void Close();
```

Exceptions

InvalidOperationException - The Close method was called from a OracleRowsCopied event.

Remarks

After the Close method is called on a OracleBulkCopy object, no other operation can succeed. Calls to the WriteToServer method throw an InvalidOperationException. The Close method closes the connection if the connection was opened by the OracleBulkCopy object, that is, if the OracleBulkCopy object was created by a constructor that takes a connection string.

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

Dispose

This method releases any resources or memory allocated by the object.

Declaration

```csharp
// C#
public void Dispose();
```
Implements
IDisposable

Remarks
After the Dispose method is called on the OracleBulkCopy object, no other operation can succeed. The connection is closed if the connection was opened by the OracleBulkCopy object, that is, if a constructor that takes a connection string created the OracleBulkCopy object.

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

WriteToServer

WriteToServer copies rows to a destination table.

Overload List:
- WriteToServer(DataRow[])  
  This method copies all rows from the supplied DataRow array to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.
- WriteToServer(DataTable)  
  This method copies all rows in the supplied DataTable to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.
- WriteToServer(IDataReader)  
  This method copies all rows in the supplied IDataReader to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.
- WriteToServer(DataTable, DataRowState)  
  This method copies rows that match the supplied row state in the supplied DataTable to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.
- WriteToServer(OracleRefCursor)  
  This method copies all rows from the specified OracleRefCursor to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
WriteToServer(DataRow[])  
This method copies all rows from the supplied DataRow array to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

Declaration  
// C#  
public void WriteToServer(DataRow[] rows);

Parameters  
- rows  
  An array of DataRow objects to be copied to the destination table.

Exceptions  
ArgumentNullException - The rows parameter is null.
InvalidOperationException - The connection is not in an open state.

Remarks  
The ColumnMappings collection maps from the DataRow columns to the destination database table.

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

WriteToServer(DataTable)  
This method copies all rows in the supplied DataTable to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

Declaration  
// C#  
public void WriteToServer(DataTable table);

Parameters  
- table  
  The source DataTable containing rows to be copied to the destination table.

Exceptions  
ArgumentNullException - The table parameter is null.
InvalidOperationException - The connection is not in an open state.

Remarks  
All rows in the DataTable are copied to the destination table except those that have been deleted.  
The ColumnMappings collection maps from the DataTable columns to the destination database table.
OracleBulkCopy Class

WriteToServer(IDataReader)
This method copies all rows in the supplied IDataReader to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

Declaration
// C#
public void WriteToServer(IDataReader reader);

Parameters
■ reader
A IDataReader instance containing rows to be copied to the destination table.

Exceptions
ArgumentNullException - The reader parameter is null.
InvalidOperationException - The connection is not in an open state.

Remarks
The bulk copy operation starts with the next available row of the data reader. Typically, the reader returned by a call to the ExecuteReader method is passed to the WriteToServer method so that the next row becomes the first row. To copy multiple result sets, the application must call NextResult on the reader and then call the WriteToServer method again.

This WriteToServer method changes the state of the reader as it calls reader.Read internally to get the source rows. Thus, at the end of the WriteToServer operation, the reader is at the end of the result set.

The ColumnMappings collection maps from the data reader columns to the destination database table.

See Also:
■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleBulkCopy Class
■ OracleBulkCopy Members

WriteToServer(DataTable, DataRowState)
This method copies rows that match the supplied row state in the supplied DataTable to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

Declaration
// C#
public void WriteToServer(DataTable table, DataRowState rowState);

See Also:
■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleBulkCopy Class
■ OracleBulkCopy Members
Parameters

- **table**
  A DataTable containing rows to be copied to the destination table.

- **rowState**
  The DataRowState enumeration value. Only rows matching the row state are copied to the destination.

Exceptions

- ArgumentException - The table or rowState parameter is null.
- InvalidOperationException - The connection is not in an open state.

Remarks

Only rows in the DataTable that are in the state indicated in the rowState argument and have not been deleted are copied to the destination table.

The ColumnMappings collection maps from the DataTable columns to the destination database table.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members

WriteToServer(OracleRefCursor)

This method copies all rows from the specified OracleRefCursor to a destination table specified by the DestinationTableName property of the OracleBulkCopy object.

Declaration

```csharp
// C#
public void WriteToServer(OracleRefCursor refCursor);
```

Parameters

- **refCursor**
  An OracleRefCursor object containing rows to be copied to the destination table.

Exceptions

- ArgumentException - The refCursor parameter is null
- InvalidOperationException - The connection is not in an open state.

Remarks

The ColumnMappings collection maps from the OracleRefCursor columns to the destination database table.
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
OracleBulkCopy Events

OracleBulkCopy events are listed in Table 17–7.

Table 17–7 OracleBulkCopy Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleRowsCopied</td>
<td>Triggered every time the number of rows specified by the OracleBulkCopy.NotifyAfter property has been processed</td>
</tr>
</tbody>
</table>

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

OracleRowsCopied

This event is triggered every time the number of rows specified by the OracleBulkCopy.NotifyAfter property has been processed.

Declaration

```
// C#
public event OracleRowsCopiedEventHandler OracleRowsCopied;
```

Exceptions

InvalidOperationException - The Close method is called inside this event.

Remarks

This event is raised when the number of rows specified by the NotifyAfter property has been processed. It does not imply that the rows have been sent to the database or committed.

To cancel the operation from this event, use the Abort property of OracleRowsCopiedEventArgs class.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy Class
- OracleBulkCopy Members
- "NotifyAfter" on page 17-13
OracleBulkCopyColumnMapping Class

The OracleBulkCopyColumnMapping class defines the mapping between a column in the data source and a column in the destination database table.

Class Inheritance
System.Object
    System.OracleBulkCopyColumnMapping

Declaration
// C#
public sealed class OracleBulkCopyColumnMapping

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

Remarks
Column mappings define the mapping between data source and the target table.
It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

---

Note: Oracle Data Provider for .NET makes one or more round-trips to the database to determine the column name if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.

---

Example
// C#

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMapping Members
- OracleBulkCopyColumnMapping Constructors
- OracleBulkCopyColumnMapping Properties
OracleBulkCopyColumnMapping Members

OracleBulkCopyColumnMapping members are listed in the following tables.

OracleBulkCopyColumnMapping Constructors

The OracleBulkCopyColumnMapping constructors are listed in Table 17–8.

Table 17–8  OracleBulkCopyColumnMapping Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleBulkCopyColumnMapping Constructors</td>
<td>Instantiates new instances of the OracleBulkCopyColumnMapping class</td>
</tr>
</tbody>
</table>

OracleBulkCopyColumnMapping Properties

The OracleBulkCopyColumnMapping properties are listed in Table 17–9.

Table 17–9  OracleBulkCopyColumnMapping Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestinationColumn</td>
<td>Specifies the column name of the destination table that is being mapped</td>
</tr>
<tr>
<td>DestinationOrdinal</td>
<td>Specifies the column ordinal value of the destination table that is being mapped</td>
</tr>
<tr>
<td>SourceColumn</td>
<td>Specifies the column name of the data source that is being mapped</td>
</tr>
<tr>
<td>SourceOrdinal</td>
<td>Specifies the column ordinal value of the data source that is being mapped</td>
</tr>
</tbody>
</table>

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMapping Class
OracleBulkCopyColumnMapping Constructors

OracleBulkCopyColumnMapping constructors instantiates new instances of the OracleBulkCopyColumnMapping class.

Overload List:

■ **OracleBulkCopyColumnMapping()**
  This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class.

■ **OracleBulkCopyColumnMapping(int, int)**
  This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column ordinal and destination column ordinal.

■ **OracleBulkCopyColumnMapping(int, string)**
  This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column ordinal and destination column name.

■ **OracleBulkCopyColumnMapping(string, int)**
  This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column name and destination column ordinal.

■ **OracleBulkCopyColumnMapping(string, string)**
  This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column name and destination column name.

See Also:

■ "Oracle.DataAccess.Client Namespace" on page 1-4
■ OracleBulkCopyColumnMapping Class
■ OracleBulkCopyColumnMapping Members

OracleBulkCopyColumnMapping()

This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class.

Declaration

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

Remarks

Applications that use this constructor must define the source for the mapping using the SourceColumn or SourceOrdinal property, and must define the destination for the mapping using the DestinationColumn or DestinationOrdinal property.
OracleBulkCopyColumnMapping Constructors

**OracleBulkCopyColumnMapping(int, int)**

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class using the provided source and destination column ordinal positions.

**Declaration**

```csharp
// C#
public OracleBulkCopyColumnMapping(int sourceColumnOrdinal, int destinationOrdinal);
```

**Parameters**

- `sourceColumnOrdinal`
  The ordinal position of the source column within the data source.

- `destinationOrdinal`
  The ordinal position of the destination column within the destination table.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMapping Class
- OracleBulkCopyColumnMapping Members

**OracleBulkCopyColumnMapping(int, string)**

This constructor instantiates a new instance of the `OracleBulkCopyColumnMapping` class using the provided source column ordinal and destination column name.

**Declaration**

```csharp
// C#
public OracleBulkCopyColumnMapping(int sourceColumnOrdinal, string destinationColumn);
```

**Parameters**

- `sourceColumnOrdinal`
  The ordinal position of the source column within the data source.

- `destinationColumn`
  The name of the destination column within the destination table.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMapping Class
- OracleBulkCopyColumnMapping Members
OracleBulkCopyColumnMapping Class

OracleBulkCopyColumnMapping(string, int)
This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source column name and destination column ordinal.

Declaration
// C#
public OracleBulkCopyColumnMapping(string sourceColumn, int destinationOrdinal);

Parameters
- sourceColumn
  The name of the source column within the data source.
- destinationOrdinal
  The ordinal position of the destination column within the destination table.

OracleBulkCopyColumnMapping(string, string)
This constructor instantiates a new instance of the OracleBulkCopyColumnMapping class using the provided source and destination column names.

Declaration
// C#
public OracleBulkCopyColumnMapping(string sourceColumn, string destinationColumn);

Parameters
- sourceColumn
  The name of the source column within the data source.
- destinationColumn
  The name of the destination column within the destination table.

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

Oracle Data Provider for .NET Bulk Copy Classes  17-25
OracleBulkCopyColumnMapping Properties

The OracleBulkCopyColumnMapping properties are listed in Table 17–10.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestinationColumn</td>
<td>Specifies the column name of the destination table that is being mapped.</td>
</tr>
<tr>
<td>DestinationOrdinal</td>
<td>Specifies the column ordinal value of the destination table that is being mapped.</td>
</tr>
<tr>
<td>SourceColumn</td>
<td>Specifies the column name of the data source that is being mapped.</td>
</tr>
<tr>
<td>SourceOrdinal</td>
<td>Specifies the column ordinal value of the data source that is being mapped.</td>
</tr>
</tbody>
</table>

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

DestinationColumn

This property specifies the column name of the destination table that is being mapped.

Declaration

```csharp
// C#
public string DestinationColumn {get; set;}
```

Property Value

A string value that represents the destination column name of the mapping.

Remarks

The DestinationColumn and DestinationOrdinal properties are mutually exclusive. The last value set takes precedence.

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

DestinationOrdinal

This property specifies the column ordinal value of the destination table that is being mapped.

Declaration

```csharp
// C#
public int DestinationOrdinal {get; set;}
```
Property Value
An integer value that represents the destination column ordinal of the mapping.

Exceptions
IndexOutOfRangeException - The destination ordinal is invalid.

Remarks
The DestinationOrdinal and DestinationColumn properties are mutually exclusive. The last value set takes precedence.

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleBulkCopyColumnMapping Class
- OracleBulkCopyColumnMapping Members

SourceColumn
This property specifies the column name of the data source that is being mapped.

Declaration
// C#
public string SourceColumn {get; set;}

Property Value
A string value that represents the source column name of the mapping.

Remarks
The SourceColumn and SourceOrdinal properties are mutually exclusive. The last value set takes precedence.

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleBulkCopyColumnMapping Class
- OracleBulkCopyColumnMapping Members

SourceOrdinal
This property specifies the column ordinal value of the data source that is being mapped.

Declaration
// C#
public int SourceOrdinal {get; set;}

Property Value
An integer value that represents the source column ordinal of the mapping.

Exceptions
IndexOutOfRangeException - The source ordinal is invalid.
Remarks

The SourceOrdinal and SourceColumn properties are mutually exclusive. The last value set takes precedence.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMapping Class
- OracleBulkCopyColumnMapping Members
OracleBulkCopyColumnMappingCollection Class

The OracleBulkCopyColumnMappingCollection class represents a collection of OracleBulkCopyColumnMapping objects that are used to map columns in the data source to columns in a destination table.

Class Inheritance
System.Object
    System.CollectionBase
        System.OracleBulkCopyColumnMappingCollection

Declaration
// C#
public sealed class OracleBulkCopyColumnMappingCollection : CollectionBase

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

Remarks
Column mappings define the mapping between data source and the target table.

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

Note: Oracle Data Provider for .NET makes one or more round-trips to the database to determine the column name if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.

Example
// C#

Requirements
Namespace: Oracle.DataAccess.Client
Assembly: Oracle.DataAccess.dll
ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4
See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMappingCollection Class
- OracleBulkCopyColumnMappingCollection Members
- OracleBulkCopyColumnMappingCollection Properties
- OracleBulkCopyColumnMappingCollection Public Methods
OracleBulkCopyColumnMappingCollection Members

OracleBulkCopyColumnMappingCollection members are listed in the following tables.

OracleBulkCopyColumnMappingCollection Properties
The OracleBulkCopyColumnMappingCollection properties are listed in Table 17–11.

Table 17–11 OracleBulkCopyColumnMappingCollection Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item[index]</td>
<td>Gets or sets the OracleBulkCopyColumnMappingCollection object at the specified index</td>
</tr>
</tbody>
</table>

OracleBulkCopyColumnMappingCollection Public Methods
The OracleBulkCopyColumnMappingCollection public methods are listed in Table 17–12.

Table 17–12 OracleBulkCopyColumnMappingCollection 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</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the contents of the collection</td>
</tr>
<tr>
<td>Contains</td>
<td>Returns a value indicating whether or not a specified OracleBulkCopyColumnMapping object exists in the collection</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies the elements of the OracleBulkCopyColumnMappingCollection to an array of OracleBulkCopyColumnMapping items, starting at a specified index</td>
</tr>
<tr>
<td>IndexOf</td>
<td>Returns the index of the specified OracleBulkCopyColumnMapping object</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts a new OracleBulkCopyColumnMapping object in the collection, at the index specified.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes the specified OracleBulkCopyColumnMapping element from the OracleBulkCopyColumnMappingCollection.</td>
</tr>
<tr>
<td>RemoveAt</td>
<td>Removes the mapping from the collection at the specified index.</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMappingCollection Class
OracleBulkCopyColumnMappingCollection Properties

The OracleBulkCopyColumnMappingCollection properties are listed in Table 17–13.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item[index]</td>
<td>Gets or sets the OracleBulkCopyColumnMappingCollection object at the specified index</td>
</tr>
</tbody>
</table>

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

Item[index]

This property gets or sets the OracleBulkCopyColumnMapping object at the specified index.

Declaration

```csharp
// C#
public OracleBulkCopyColumnMapping this[int index] {get;set;}
```

Parameters

- index

  The zero-based index of the OracleBulkCopyColumnMapping being set or retrieved.

Property Value

An OracleBulkCopyColumnMapping object at the specified index.

Exceptions

IndexOutOfRange Exception - The specified index does not exist.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMappingCollection Class
- OracleBulkCopyColumnMappingCollection Members
OracleBulkCopyColumnMappingCollection Public Methods

The OracleBulkCopyColumnMappingCollection public methods are listed in Table 17–14.

**Table 17–14  OracleBulkCopyColumnMappingCollection 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</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the contents of the collection</td>
</tr>
<tr>
<td>Contains</td>
<td>Returns a value indicating whether or not a specified OracleBulkCopyColumnMapping object exists in the collection</td>
</tr>
<tr>
<td>CopyTo</td>
<td>Copies the elements of the OracleBulkCopyColumnMappingCollection to an array of OracleBulkCopyColumnMapping items, starting at a specified index</td>
</tr>
<tr>
<td>IndexOf</td>
<td>Returns the index of the specified OracleBulkCopyColumnMapping object</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts a new OracleBulkCopyColumnMapping object in the collection, at the index specified.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes the specified OracleBulkCopyColumnMapping element from the OracleBulkCopyColumnMappingCollection.</td>
</tr>
<tr>
<td>RemoveAt</td>
<td>Removes the mapping from the collection at the specified index.</td>
</tr>
</tbody>
</table>

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

**Add**

Add methods add objects to the collection.

**Overload List:**
- **Add(OracleBulkCopyColumnMapping)**
  This method adds the supplied OracleBulkCopyColumnMapping object to the collection.
- **Add(int, int)**
  This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source and destination column ordinal positions.
- **Add(int, string)**
  This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source column ordinal and destination column name.
- **Add(string, int)**
  This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source column name and destination column ordinal.
Add(string, string)
This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source and destination column names.

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

Add(OracleBulkCopyColumnMapping)
This method adds the supplied OracleBulkCopyColumnMapping object to the collection.

Declaration
// C#
public OracleBulkCopyColumnMapping Add(OracleBulkCopyColumnMapping bulkCopyColumnMapping);

Parameters
- bulkCopyColumnMapping
  The OracleBulkCopyColumnMapping object that describes the mapping to be added to the collection.

Exceptions
InvalidOperationException - The bulk copy operation is in progress.

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

Add(int, int)
This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source and destination column ordinal positions.

Declaration
// C#
public OracleBulkCopyColumnMapping Add(int sourceColumnIndex, int destinationColumnIndex);

Parameters
- sourceColumnIndex
  The ordinal position of the source column within the data source.
- destinationColumnIndex
  The ordinal position of the destination column within the destination table.
Exceptions

InvalidOperationException - The bulk copy operation is in progress.

Return Value

The newly created `OracleBulkCopyColumnMapping` object that was added to the collection.

Remarks

It is not necessary to specify column mappings for all the columns in the data source. If a `ColumnMapping` is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an `InvalidOperationException` is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

---

**Note:** Oracle Data Provider for .NET makes one or more round-trips to the database to determine the column name if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.

---

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleBulkCopyColumnMappingCollection` Class
- `OracleBulkCopyColumnMappingCollection` Members

---

**Add(int, string)**

This method creates and adds an `OracleBulkCopyColumnMapping` object to the collection using the supplied source column ordinal and destination column name.

Declaration

```csharp
public OracleBulkCopyColumnMapping Add(int sourceColumnIndex,
                                         string destinationColumn);
```

Parameters

- `sourceColumnIndex`
  - The ordinal position of the source column within the data source.

- `destinationColumn`
  - The name of the destination column within the destination table.

Exceptions

InvalidOperationException - The bulk copy operation is in progress.

Return Value

The newly created `OracleBulkCopyColumnMapping` object that was added to the collection.
Remarks

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

---

Note: Oracle Data Provider for .NET makes one or more round trips to the database to determine the column names if the mapping is specified by ordinal resulting in a performance overhead. Therefore, it is recommended to specify the mapping by column names.

---

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

Add(string, int)

This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source column name and destination column ordinal.

Declaration

```
// C#
public OracleBulkCopyColumnMapping Add(string sourceColumn,
   int destinationColumnIndex);
```

Parameters

- `sourceColumn`
  The name of the source column within the data source.

- `destinationColumnIndex`
  The ordinal position of the destination column within the destination table.

Return Value

The newly created OracleBulkCopyColumnMapping object that was added to the collection.

Exceptions

- `InvalidOperationException` - The bulk copy operation is in progress.

Remarks

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.
Add(string, string)

This method creates and adds an OracleBulkCopyColumnMapping object to the collection using the supplied source and destination column names.

Declaration

// C#
public OracleBulkCopyColumnMapping Add(string sourceColumn,
    string destinationColumn);

Parameters

- sourceColumn
  The name of the source column within the data source.
- destinationColumn
  The name of the destination column within the destination table.

Exceptions

InvalidOperationException - The bulk copy operation is in progress.

Return Value

The newly created OracleBulkCopyColumnMapping object that was added to the collection.

Remarks

It is not necessary to specify column mappings for all the columns in the data source. If a ColumnMapping is not specified, then, by default, columns are mapped based on the ordinal position. This succeeds only if the source and destination table schema match. If there is a mismatch, an InvalidOperationException is thrown.

All the mappings in a mapping collection must be by name or ordinal position.

Note: Oracle Data Provider for .NET makes one or more round-trips to the database to determine the column names if the mapping is specified by ordinal position. To avoid this performance overhead, specify the mapping by column name.
### Clear

This method clears the contents of the collection.

**Declaration**

```csharp
// C#
public void Clear();
```

**Exceptions**

`InvalidOperationException` - The bulk copy operation is in progress.

**Remarks**

The `Clear` method is most commonly used when an application uses a single `OracleBulkCopy` instance to process more than one bulk copy operation. If column mappings are created for one bulk copy operation, the `OracleBulkCopyColumnMappingCollection` must be cleared after the `WriteToServer` method invocation and before the next bulk copy is processed.

It is usually more efficient to perform several bulk copies using the same `OracleBulkCopy` instance than to use a separate `OracleBulkCopy` for each operation.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleBulkCopyColumnMappingCollection Class`
- `OracleBulkCopyColumnMappingCollection Members`

### Contains

This method returns a value indicating whether or not a specified `OracleBulkCopyColumnMapping` object exists in the collection.

**Declaration**

```csharp
// C#
public bool Contains(OracleBulkCopyColumnMapping value);
```

**Parameters**

- `value` - A valid `OracleBulkCopyColumnMapping` object.

**Return Value**

Returns `true` if the specified mapping exists in the collection; otherwise, returns `false`.

**See Also:**

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleBulkCopyColumnMappingCollection Class`
- `OracleBulkCopyColumnMappingCollection Members`
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMappingCollection Class
- OracleBulkCopyColumnMappingCollection Members

**CopyTo**

This method copies the elements of the OracleBulkCopyColumnMappingCollection to an array of OracleBulkCopyColumnMapping items, starting at a specified index.

**Declaration**

```csharp
// C#
public void CopyTo(OracleBulkCopyColumnMapping[] array, int index);
```

**Parameters**

- `array`

  The one-dimensional OracleBulkCopyColumnMapping array that is the destination for the elements copied from the OracleBulkCopyColumnMappingCollection object. The array must have zero-based indexing.

- `index`

  The zero-based array index at which copying begins.

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

**IndexOf**

This method returns the index of the specified OracleBulkCopyColumnMapping object.

**Declaration**

```csharp
// C#
public int IndexOf(OracleBulkCopyColumnMapping value);
```

**Parameters**

- `value`

  The OracleBulkCopyColumnMapping object that is being returned.

**Return Value**

The zero-based index of the column mapping or -1 if the column mapping is not found in the collection.
See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopyColumnMappingCollection Class
- OracleBulkCopyColumnMappingCollection Members

Insert

This method inserts a new OracleBulkCopyColumnMapping object in the collection, at the index specified.

Declaration

// C#
public void Insert(int index, OracleBulkCopyColumnMapping value);

Parameters

- **index**
  The integer value of the location within the OracleBulkCopyColumnMappingCollection at which the new OracleBulkCopyColumnMapping is inserted.
- **value**
  The OracleBulkCopyColumnMapping object to be inserted in the collection.

Exceptions

InvalidOperationException - The bulk copy operation is in progress.

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

Remove

This method removes the specified OracleBulkCopyColumnMapping element from the OracleBulkCopyColumnMappingCollection.

Declaration

// C#
public void Remove(OracleBulkCopyColumnMapping value);

Parameters

- **value**
  The OracleBulkCopyColumnMapping object to be removed from the collection.

Exceptions

InvalidOperationException - The bulk copy operation is in progress.
Remarks

The `Remove` method is most commonly used when a single `OracleBulkCopy` instance processes more than one bulk copy operation. If column mappings are created for one bulk copy operation, mappings that no longer apply must be removed after the `WriteToServer` method invocation and before mappings are defined for the next bulk copy. The `Clear` method can clear the entire collection, and the `Remove` and the `RemoveAt` methods can remove mappings individually.

It is usually more efficient to perform several bulk copies using the same `OracleBulkCopy` instance than to use a separate `OracleBulkCopy` for each operation.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleBulkCopyColumnMappingCollection` Class
- `OracleBulkCopyColumnMappingCollection` Members

RemoveAt

This method removes the mapping from the collection at the specified index.

Declaration

// C#
public void RemoveAt(int index);

Parameters

- `index`
  The zero-based index of the `OracleBulkCopyColumnMapping` object to be removed from the collection.

Exceptions

`InvalidOperationException` - The bulk copy operation is in progress.

Remarks

The `RemoveAt` method is most commonly used when a single `OracleBulkCopy` instance is used to process more than one bulk copy operation. If column mappings are created for one bulk copy operation, mappings that no longer apply must be removed after the `WriteToServer` method invocation and before the mappings for the next bulk copy are defined. The `Clear` method can clear the entire collection, and the `Remove` and the `RemoveAt` methods can remove mappings individually.

It is usually more efficient to perform several bulk copies using the same `OracleBulkCopy` instance than to use a separate `OracleBulkCopy` for each operation.

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- `OracleBulkCopyColumnMappingCollection` Class
- `OracleBulkCopyColumnMappingCollection` Members
OracleBulkCopyOptions Enumeration

The OracleBulkCopyOptions enumeration specifies the values that can be combined with an instance of the OracleBulkCopy class and used as options to determine its behavior and the behavior of the WriteToServer methods for that instance.

Table 17–15 lists all the OracleBulkCopyOptions enumeration values with a description of each enumerated value.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Indicates that the default value for all options are to be used.</td>
</tr>
<tr>
<td>UseInternalTransaction</td>
<td>Indicates that each batch of the bulk copy operation occurs within a transaction. If the connection used to perform the bulk copy operation is already part of a transaction, an InvalidOperationException exception is raised. If this member is not specified, BatchSize number of rows are sent to the database, without any transaction-related activity.</td>
</tr>
</tbody>
</table>

**Note:** All bulk copy operations are agnostic of any local or distributed transaction created by the application.

Requirements

Namespace: Oracle.DataAccess.Client

Assembly: Oracle.DataAccess.dll

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleBulkCopy "BulkCopyOptions" on page 17-10
OracleRowsCopiedEventHandler Delegate

The OracleRowsCopiedEventHandler delegate represents the method that handles the OracleRowsCopied event of an OracleBulkCopy object.

Declaration

// C#
public delegate void OracleRowsCopiedEventHandler (object sender,
            OracleRowsCopiedEventArgs eventArgs);

Parameters

- **sender**
  The source of the event.

- **eventArgs**
  The OracleRowsCopiedEventArgs object that contains the event data.

Remarks

Event callbacks can be registered through this event delegate for applications that wish to be notified every time the number of rows specified by the OracleBulkCopy.NotifyAfter property has been processed.

If the event handler calls the OracleBulkCopy.Close method, an exception is generated, and the OracleBulkCopy object state does not change.

The event handler can also set the OracleRowsCopiedEventArgs.Abort property to true to indicate that the bulk copy operation must be aborted. If the bulk copy operation is part of an external transaction, an exception is generated and the transaction is not rolled back. The application is responsible for either committing or rolling back the external transaction.

If there is no external transaction, the internal transaction for the current batch of rows is automatically rolled back. However the previous batches of imported rows are unaffected, as their transactions have already been committed.

Requirements

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

ODP.NET Version: ODP.NET for .NET Framework 2.0 or ODP.NET for .NET Framework 4

See Also:

- "Oracle.DataAccess.Client Namespace" on page 1-4
- "OracleRowsCopied" on page 20
- "NotifyAfter" on page 17-13
OracleRowsCopiedEventArgs Class

The OracleRowsCopiedEventArgs class represents the set of arguments passed as part of event data for the OracleRowsCopied event.

Class Inheritance
System.Object
   System.EventArgs
      System.OracleRowsCopiedEventArgs

Declaration
// C#
public class OracleRowsCopiedEventArgs : EventArgs

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

Remarks
Each time the number of rows represented by the OracleBulkCopy.NotifyAfter property is processed, the OracleBulkCopy.OracleRowsCopied event is raised, providing an OracleRowsCopiedEventArgs object that stores the event data.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleRowsCopiedEventArgs Members
- OracleRowsCopiedEventArgs Constructors
- OracleRowsCopiedEventArgs Properties
OracleRowsCopiedEventArgs Members

OracleRowsCopiedEventArgs members are listed in the following tables.

**OracleRowsCopiedEventArgs Constructors**

OracleRowsCopiedEventArgs constructors are listed in Table 17–16.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleRowsCopiedEventArgs Constructors</td>
<td>OracleRowsCopiedEventArgs creates new instances of the OracleRowsCopiedEventArgs class</td>
</tr>
</tbody>
</table>

**OracleRowsCopiedEventArgs Properties**

OracleRowsCopiedEventArgs properties are listed in Table 17–17.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort</td>
<td>Retrieves or sets a value that indicates whether or not the bulk copy operation is aborted</td>
</tr>
<tr>
<td>RowsCopied</td>
<td>Retrieves a value that represents the number of rows copied during the current bulk copy operation</td>
</tr>
</tbody>
</table>

See Also:
- "Oracle.DataAccess.Client Namespace” on page 1-4
- OracleRowsCopiedEventArgs Class
OracleRowsCopiedEventArgs Constructors

OracleRowsCopiedEventArgs creates new instances of the OracleRowsCopiedEventArgs class.

Overload List:
- OracleRowsCopiedEventArgs(long)
  This constructor creates a new instance of the OracleRowsCopiedEventArgs object.

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

OracleRowsCopiedEventArgs(long)
This constructor creates a new instance of the OracleRowsCopiedEventArgs object.

Declaration
  // C#
  public OracleRowsCopiedEventArgs(long rowsCopied);

Parameters
- rowsCopied
  An Int64 value that indicates the number of rows copied during the current bulk copy operation.

Remarks
The value in the rowsCopied parameter is reset by each call to a WriteToServer method.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleRowsCopiedEventArgs Class
- OracleRowsCopiedEventArgs Members
OracleRowsCopiedEventArgs Properties

OracleRowsCopiedEventArgs properties are listed in Table 17–18.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort</td>
<td>Retrieves or sets a value that indicates whether or not the bulk copy operation is aborted</td>
</tr>
<tr>
<td>RowsCopied</td>
<td>Retrieves a value that represents the number of rows copied during the current bulk copy operation</td>
</tr>
</tbody>
</table>

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

Abort

This property retrieves or sets a value that indicates whether or not the bulk copy operation is aborted.

Declaration

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

Property Value

Returns true if the bulk copy operation is to be aborted; otherwise, returns false.

Remarks

Set the Abort property to true to cancel the bulk copy operation.

If the Close method is called from OracleRowsCopied, an exception is generated, and the OracleBulkCopy object state does not change.

If the application does not create a transaction, the internal transaction corresponding to the current batch is automatically rolled back. However, changes related to previous batches within the bulk copy operation are retained, because the transactions in those batches are committed. This case is applicable only when UseInternalTransaction bulk copy option is chosen.

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

RowsCopied

This property retrieves a value that represents the number of rows copied during the current bulk copy operation.
Declaration

// C#
public long RowsCopied {get;}

Property Value
An Int64 value that returns the number of rows copied.

Remarks
The value in the RowsCopied property is reset by each call to a WriteToServer method.

See Also:
- "Oracle.DataAccess.Client Namespace" on page 1-4
- OracleRowsCopiedEventArgs Class
- OracleRowsCopiedEventArgs Members
ODP.NET provides standard metadata collections as well as various Oracle database-specific metadata collections that can be retrieved through the OracleConnection.GetSchema API.

**See Also:**
- "Support for Schema Discovery" on page 3-23
- "GetSchema" on page 5-100

This appendix contains the following topics:
- Common Schema Collections
- ODP.NET-Specific Schema Collection

## Common Schema Collections

The common schema collections are available for all .NET Framework managed providers. ODP.NET supports the same common schema collections.

**See Also:**  "Understanding the Common Schema Collections" in the MSDN Library

- MetaDataCollections
- DataSourceInformation
- DataTypes
- Restrictions
- ReservedWords

## MetaDataCollections

*Table A–1* is a list of metadata collections that is available from the data source, such as tables, columns, indexes, and stored procedures.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CollectionName</td>
<td>string</td>
<td>The name of the collection passed to the GetSchema method for retrieval.</td>
</tr>
</tbody>
</table>
Table A–1 (Cont.) MetaDataCollections

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfRestrictions</td>
<td>int</td>
<td>Number of restrictions specified for the named collection.</td>
</tr>
<tr>
<td>NumberOfIdentifierParts</td>
<td>int</td>
<td>Number of parts in the composite identifier/database object name.</td>
</tr>
</tbody>
</table>

Table A–2  DataSourceInformation

Table A–2 lists DataSourceInformation information which may include these columns and possibly others.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompositeIdentifierSeparatorPattern</td>
<td>string</td>
<td>Separator for multipart names: @</td>
</tr>
<tr>
<td>DataSourceProductName</td>
<td>string</td>
<td>Database name: Oracle</td>
</tr>
<tr>
<td>DataSourceProductVersion</td>
<td>string</td>
<td>Database version. Note that this is the version of the database instance currently being accessed by DbConnection.</td>
</tr>
<tr>
<td>DataSourceProductVersionNormalized</td>
<td>string</td>
<td>A normalized DataSource version for easier comparison between different versions. For example: DataSource Version: 10.2.0.1.0 Normalized DataSource Version: 10.02.00.01.00</td>
</tr>
<tr>
<td>GroupByBehavior</td>
<td>GroupByBehavior</td>
<td>An enumeration that indicates the relationship between the columns in a GROUP BY clause and the non-aggregated columns in a select list.</td>
</tr>
<tr>
<td>IdentifierPattern</td>
<td>string</td>
<td>Format for a valid identifier.</td>
</tr>
<tr>
<td>IdentifierCase</td>
<td>IdentifierCase</td>
<td>An enumeration that specifies whether or not to treat non-quoted identifiers as case sensitive.</td>
</tr>
<tr>
<td>OrderByColumnsInSelect</td>
<td>bool</td>
<td>A boolean that indicates whether or not the select list must contain the columns in an ORDER BY clause.</td>
</tr>
<tr>
<td>ParameterMarkerFormat</td>
<td>string</td>
<td>A string indicating whether or not parameter markers begin with a special character.</td>
</tr>
<tr>
<td>ParameterMarkerPattern</td>
<td>string</td>
<td>The format of a parameter marker.</td>
</tr>
<tr>
<td>ParameterNameMaxLength</td>
<td>int</td>
<td>Maximum length of a parameter.</td>
</tr>
<tr>
<td>ParameterNamePattern</td>
<td>string</td>
<td>The format for a valid parameter name.</td>
</tr>
<tr>
<td>QuotedIdentifierPattern</td>
<td>string</td>
<td>The format of a quoted identifier.</td>
</tr>
<tr>
<td>QuotedIdentifierCase</td>
<td>IdentifierCase</td>
<td>An enumeration that specifies whether or not to treat quote identifiers as case sensitive.</td>
</tr>
<tr>
<td>StringLiteralPattern</td>
<td>string</td>
<td>The format for a string literal.</td>
</tr>
<tr>
<td>SupportedJoinOperators</td>
<td>SupportedJoin Operators</td>
<td>An enumeration indicating the types of SQL join statements supported by the data source.</td>
</tr>
</tbody>
</table>
**DataTypes**

Table A–3 lists DataTypes Collection information which may include these columns and possibly others.

---

**Note:** As an example, the description column includes complete information for the TIMESTAMP WITH LOCAL TIME ZONE data type.

---

**Table A–3  DataTypes**

<table>
<thead>
<tr>
<th>ColumnName</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeName</td>
<td>string</td>
<td>The provider-specific data type name. Example: TIMESTAMP WITH LOCAL TIME ZONE</td>
</tr>
<tr>
<td>ProviderDbType</td>
<td>int</td>
<td>The provider-specific type value. Example: 124</td>
</tr>
<tr>
<td>ColumnSize</td>
<td>long</td>
<td>The length of a non-numeric column or parameter. Example: 27</td>
</tr>
<tr>
<td>CreateFormat</td>
<td>string</td>
<td>A format string that indicates how to add this column to a DDL statement.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: TIMESTAMP({0} WITH LOCAL TIME ZONE)</td>
</tr>
<tr>
<td>CreateParameters</td>
<td>string</td>
<td>The parameters specified to create a column of this data type. Example: 8</td>
</tr>
<tr>
<td>DataType</td>
<td>string</td>
<td>The .NET type for the data type. Example: System.DateTime</td>
</tr>
<tr>
<td>IsAutoIncrementable</td>
<td>bool</td>
<td>A boolean value that indicates whether or not this data type can be auto-incremented. Example: false</td>
</tr>
<tr>
<td>IsBestMatch</td>
<td>bool</td>
<td>A boolean value that indicates whether or not this data type is the best match to values in the DataType column. Example: false</td>
</tr>
<tr>
<td>IsCaseSensitive</td>
<td>bool</td>
<td>A boolean value that indicates whether or not this data type is case-sensitive. Example: false</td>
</tr>
<tr>
<td>IsFixedLength</td>
<td>bool</td>
<td>A boolean value that indicates whether or not this data type has a fixed length. Example: true</td>
</tr>
<tr>
<td>IsFixedPrecisionScale</td>
<td>bool</td>
<td>A boolean value that indicates whether or not this data type has a fixed precision and scale. Example: false</td>
</tr>
<tr>
<td>IsLong</td>
<td>bool</td>
<td>A boolean value that indicates whether or not this data type contains very long data. Example: false</td>
</tr>
</tbody>
</table>
Table A–4  Restrictions

<table>
<thead>
<tr>
<th>ColumnName</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CollectionName</td>
<td>string</td>
<td>The collection that the restrictions apply to.</td>
</tr>
<tr>
<td>RestrictionName</td>
<td>string</td>
<td>The restriction name.</td>
</tr>
<tr>
<td>RestrictionNumber</td>
<td>int</td>
<td>A number that indicates the location of the restriction.</td>
</tr>
</tbody>
</table>

Restrictions

Table A–4 lists Restrictions, including the following columns.
ReservedWords

The `ReservedWords` collection exposes information about the words that are reserved by the database currently connected to ODP.NET.

*Table A–5* lists the `ReservedWords` Collection.

<table>
<thead>
<tr>
<th>ColumnName</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReservedWord</td>
<td>string</td>
<td>Provider-specific reserved words</td>
</tr>
</tbody>
</table>

**ODP.NET-Specific Schema Collection**

Oracle Data Provider for .NET supports both the common schema collections described previously and the following Oracle-specific schema collections:

- Tables
- Columns
- Views
- XMLSchema
- Users
- Synonyms
- Sequences
- Functions
- Procedures
- ProcedureParameters
- Arguments
- Packages
- PackageBodies
- JavaClasses
- Indexes
- IndexColumns
- PrimaryKeys
- ForeignKeys
- ForeignKeyColumns
- UniqueKeys

**Tables**

*Table A–6* lists the column name, data type, and description of the Tables Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the Table.</td>
</tr>
</tbody>
</table>
Columns

Table A–7 lists the column name, data type, and description of the Columns Schema Collection.

Table A–7 Columns

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the table or view.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>String</td>
<td>Name of the table or view.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>String</td>
<td>Name of the column.</td>
</tr>
<tr>
<td>ID</td>
<td>Decimal</td>
<td>Sequence number of the column as created.</td>
</tr>
<tr>
<td>DATATYPE</td>
<td>String</td>
<td>Data type of the column.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>Decimal</td>
<td>Length of the column in bytes.</td>
</tr>
<tr>
<td>PRECISION</td>
<td>Decimal</td>
<td>Decimal precision for NUMBER data type; binary precision for FLOAT data type, null for all other data types.</td>
</tr>
<tr>
<td>Scale</td>
<td>Decimal</td>
<td>Digits to right of decimal point in a number.</td>
</tr>
<tr>
<td>NULLABLE</td>
<td>String</td>
<td>Specifies whether or not a column allows NULLs.</td>
</tr>
<tr>
<td>CHAR_USED</td>
<td>String</td>
<td>Indicates whether the column uses BYTE length semantics (B) or CHAR length semantics (C).</td>
</tr>
<tr>
<td>LengthInChars</td>
<td>Decimal</td>
<td>Length of the column in characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This value only applies to CHAR, VARCHAR2, NCHAR, and NVARCHAR2.</td>
</tr>
</tbody>
</table>

Views

Table A–8 lists the column name, data type, and description of the Views Schema Collection.

Table A–8 Views

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the view.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>String</td>
<td>Name of the view.</td>
</tr>
<tr>
<td>TEXT_LENGTH</td>
<td>Decimal</td>
<td>Length of the view text.</td>
</tr>
<tr>
<td>TEXT</td>
<td>String</td>
<td>View text.</td>
</tr>
<tr>
<td>TYPE_TEXT_LENGTH</td>
<td>Decimal</td>
<td>Length of the type clause of the typed view.</td>
</tr>
<tr>
<td>TYPE_TEXT</td>
<td>String</td>
<td>Type clause of the typed view.</td>
</tr>
<tr>
<td>OID_TEXT_LENGTH</td>
<td>Decimal</td>
<td>Length of the WITH OID clause of the typed view.</td>
</tr>
<tr>
<td>OID_TEXT</td>
<td>String</td>
<td>WITH OID clause of the typed view.</td>
</tr>
<tr>
<td>VIEW_TYPE_OWNER</td>
<td>String</td>
<td>Owner of the view type if the view is a typed view.</td>
</tr>
</tbody>
</table>
XMLSchema

Table A–9 lists the column name, data type and description of the XMLSchema Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the XML schema.</td>
</tr>
<tr>
<td>SCHEMA_URL</td>
<td>String</td>
<td>Schema URL of the XML schema.</td>
</tr>
<tr>
<td>LOCAL</td>
<td>String</td>
<td>Indicates whether the XML schema is local (YES) or global (NO).</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>String</td>
<td>XML schema document.</td>
</tr>
<tr>
<td>INT_OBJNAME</td>
<td>String</td>
<td>Internal database object name for the schema.</td>
</tr>
<tr>
<td>QUAL_SCHEMA_URL</td>
<td>String</td>
<td>Fully qualified schema URL.</td>
</tr>
<tr>
<td>HIER_TYPE</td>
<td>String</td>
<td>Hierarchy type for the schema.</td>
</tr>
</tbody>
</table>

Users

Table A–10 lists the column name, data type and description of the Users Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>String</td>
<td>Name of the user.</td>
</tr>
<tr>
<td>ID</td>
<td>Decimal</td>
<td>ID number of the user.</td>
</tr>
<tr>
<td>CREATEDATE</td>
<td>DateTime</td>
<td>User creation date.</td>
</tr>
</tbody>
</table>

Synonyms

Table A–11 lists the column name, data type and description of the Synonyms Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the synonym.</td>
</tr>
<tr>
<td>SYNONYM_NAME</td>
<td>String</td>
<td>Name of the synonym.</td>
</tr>
</tbody>
</table>
Sequences

Table A–12 lists the column name, data type, and description of the Sequences Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQUENCE_OWNER</td>
<td>String</td>
<td>Name of the owner of the sequence.</td>
</tr>
<tr>
<td>SEQUENCE_NAME</td>
<td>String</td>
<td>Sequence name.</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>Decimal</td>
<td>Minimum value of the sequence.</td>
</tr>
<tr>
<td>MAX_VALUE</td>
<td>Decimal</td>
<td>Maximum value of the sequence.</td>
</tr>
<tr>
<td>INCREMENT_BY</td>
<td>Decimal</td>
<td>Value by which sequence is incremented.</td>
</tr>
<tr>
<td>CYCLE_FLAG</td>
<td>String</td>
<td>Indicates if sequence wraps around on reaching limit.</td>
</tr>
<tr>
<td>ORDER_FLAG</td>
<td>String</td>
<td>Indicates if sequence numbers are generated in order.</td>
</tr>
<tr>
<td>CACHE_SIZE</td>
<td>Decimal</td>
<td>Number of sequence numbers to cache.</td>
</tr>
<tr>
<td>LAST_NUMBER</td>
<td>Decimal</td>
<td>Last sequence number written to disk. If a sequence uses caching, the number written to disk is the last number placed in the sequence cache. This number is likely to be greater than the last sequence number that was used.</td>
</tr>
</tbody>
</table>

Functions

Table A–13 lists the column name, data type, and description of the Functions Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the function.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>String</td>
<td>Name of the function.</td>
</tr>
<tr>
<td>SUBOBJECT_NAME</td>
<td>String</td>
<td>Name of the subobject (for example, partition).</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>Decimal</td>
<td>Dictionary object number of the function.</td>
</tr>
</tbody>
</table>
Table A–14  Procedures

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the procedure.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>String</td>
<td>Name of the procedure.</td>
</tr>
<tr>
<td>SUBOBJECT_NAME</td>
<td>String</td>
<td>Name of the subobject (for example, partition).</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>Decimal</td>
<td>Dictionary object number of the procedure.</td>
</tr>
<tr>
<td>DATA_OBJECT_ID</td>
<td>Decimal</td>
<td>Dictionary object number of the segment that contains the procedure.</td>
</tr>
<tr>
<td>CREATED</td>
<td>DateTime</td>
<td>Timestamp for the creation of the procedure.</td>
</tr>
<tr>
<td>LAST_DDL_TIME</td>
<td>Decimal</td>
<td>Timestamp for the last modification of the procedure resulting from a DDL statement (including grants and revokes).</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>String</td>
<td>Timestamp for the specification of the procedure (character data).</td>
</tr>
<tr>
<td>STATUS</td>
<td>String</td>
<td>Status of the procedure (VALID, INVALID, or N/A).</td>
</tr>
<tr>
<td>TEMPORARY</td>
<td>String</td>
<td>Whether or not the procedure is temporary (the current session can see only data that it placed in this object itself).</td>
</tr>
<tr>
<td>GENERATED</td>
<td>String</td>
<td>Indicates whether the name of this procedure is system generated (Y) or not (N).</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>String</td>
<td>Whether or not this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y</td>
</tr>
</tbody>
</table>

Procedures

Table A–14 lists the column name, data type, and description of the Procedures Schema Collection.
**ProcedureParameters**

Table A–15 lists the column name, data type and description of the ProcedureParameters Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>String</td>
<td>Name of the procedure or function.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>String</td>
<td>Name of the package.</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>Decimal</td>
<td>Object number of the object.</td>
</tr>
<tr>
<td>OVERLOAD</td>
<td>String</td>
<td>Indicates the nth overloading ordered by its appearance in the source; otherwise, it is NULL.</td>
</tr>
<tr>
<td>SUBPROGRAM_ID</td>
<td>Decimal</td>
<td>Subprogram id for the procedure or function</td>
</tr>
<tr>
<td>ARGUMENT_NAME</td>
<td>String</td>
<td>If the argument is a scalar type, then the argument name is the name of the argument. A null argument name is used to denote a function return value.</td>
</tr>
<tr>
<td>POSITION</td>
<td>Decimal</td>
<td>If DATA_LEVEL is zero, then this column holds the position of this item in the argument list, or zero for a function return value.</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>Decimal</td>
<td>Defines the sequential order of the argument. Argument sequence starts from 1.</td>
</tr>
<tr>
<td>DATA_LEVEL</td>
<td>Decimal</td>
<td>Nesting depth of the argument for composite types.</td>
</tr>
<tr>
<td>DATA_TYPE</td>
<td>String</td>
<td>Data type of the argument.</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>String</td>
<td>Default value for the argument.</td>
</tr>
<tr>
<td>DEFAULT_LENGTH</td>
<td>Decimal</td>
<td>Length of the default value for the argument.</td>
</tr>
<tr>
<td>IN_OUT</td>
<td>String</td>
<td>Direction of the argument: [IN] [OUT] [IN/OUT].</td>
</tr>
<tr>
<td>DATA_LENGTH</td>
<td>Decimal</td>
<td>Length of the column (in bytes).</td>
</tr>
<tr>
<td>DATA_PRECISION</td>
<td>Decimal</td>
<td>Length in decimal digits (NUMBER) or binary digits (FLOAT).</td>
</tr>
<tr>
<td>DATA_SCALE</td>
<td>Decimal</td>
<td>Digits to the right of the decimal point in a number.</td>
</tr>
<tr>
<td>RADIX</td>
<td>Decimal</td>
<td>Argument radix for a number.</td>
</tr>
<tr>
<td>CHARACTER_SET_NAME</td>
<td>String</td>
<td>Character set name for the argument.</td>
</tr>
<tr>
<td>TYPE_OWNER</td>
<td>String</td>
<td>Owner of the type of the argument.</td>
</tr>
<tr>
<td>TYPE_NAME</td>
<td>String</td>
<td>Name of the type of the argument. If the type is a package local type (that is, it is declared in a package specification), then this column displays the name of the package.</td>
</tr>
<tr>
<td>TYPE_SUBNAME</td>
<td>String</td>
<td>Displays the name of the type declared in the package identified in the TYPE_NAME column. Relevant only for package local types.</td>
</tr>
<tr>
<td>TYPE_LINK</td>
<td>String</td>
<td>Displays the database link that refers to the remote package. Relevant only for package local types when the package identified in the TYPE_NAME column is a remote package.</td>
</tr>
</tbody>
</table>
Table A–15 (Cont.) ProcedureParameters

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLS_TYPE</td>
<td>String</td>
<td>For numeric arguments, the name of the PL/SQL type of the argument. Otherwise, Null.</td>
</tr>
<tr>
<td>CHAR_LENGTH</td>
<td>Decimal</td>
<td>Character limit for string data types.</td>
</tr>
<tr>
<td>CHAR_USED</td>
<td>String</td>
<td>Indicates whether the byte limit (B) or character limit (C) is official for the string.</td>
</tr>
</tbody>
</table>

Arguments

Table A–16 lists the column name, data type, and description of the Arguments Schema Collection.

Table A–16 Arguments

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the object.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>String</td>
<td>Name of the package.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>String</td>
<td>Name of the procedure or function.</td>
</tr>
<tr>
<td>ARGUMENT_NAME</td>
<td>String</td>
<td>If the argument is a scalar type, then the argument name is the name of the argument. A null argument name is used to denote a function return value.</td>
</tr>
<tr>
<td>POSITION</td>
<td>Decimal</td>
<td>If DATA_LEVEL is zero, then this column holds the position of this item in the argument list, or zero for a function return value.</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>Decimal</td>
<td>Defines the sequential order of the argument. Argument sequence starts from 1.</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>String</td>
<td>Default value for the argument.</td>
</tr>
<tr>
<td>DEFAULT_LENGTH</td>
<td>Decimal</td>
<td>Length of the default value for the argument.</td>
</tr>
<tr>
<td>IN_OUT</td>
<td>String</td>
<td>Direction of the argument: [IN] [OUT] [IN/OUt].</td>
</tr>
<tr>
<td>DATA_LENGTH</td>
<td>Decimal</td>
<td>Length of the column (in bytes).</td>
</tr>
<tr>
<td>DATA_PRECISION</td>
<td>Decimal</td>
<td>Length in decimal digits (NUMBER) or binary digits (FLOAT).</td>
</tr>
<tr>
<td>DATA_SCALE</td>
<td>Decimal</td>
<td>Digits to the right of the decimal point in a number.</td>
</tr>
<tr>
<td>DATA_TYPE</td>
<td>String</td>
<td>Data type of the argument.</td>
</tr>
<tr>
<td>CHAR_USED</td>
<td>String</td>
<td>Indicates whether the column uses BYTE length semantics (B) or CHAR length semantics (C).</td>
</tr>
</tbody>
</table>

Packages

Table A–17 lists the column name, data type, and description of the Packages Schema Collection.

Table A–17 Packages

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the package.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>String</td>
<td>Name of the package.</td>
</tr>
<tr>
<td>SUBOBJECT_NAME</td>
<td>String</td>
<td>Name of the subobject (for example, partition).</td>
</tr>
</tbody>
</table>
### PackageBodies

Table A–18 lists the column name, data type, and description of the PackageBodies Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the package body.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>String</td>
<td>Name of the package body.</td>
</tr>
<tr>
<td>SUBOBJECT_NAME</td>
<td>String</td>
<td>Name of the subobject (for example, partition).</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>Decimal</td>
<td>Dictionary object number of the package body.</td>
</tr>
<tr>
<td>DATA_OBJECT_ID</td>
<td>Decimal</td>
<td>Dictionary object number of the segment that contains the package body.</td>
</tr>
<tr>
<td>CREATED</td>
<td>DateTime</td>
<td>Timestamp for the creation of the package body.</td>
</tr>
<tr>
<td>LAST_DDL_TIME</td>
<td>DateTime</td>
<td>Timestamp for the last modification of the package body resulting from a DDL statement (including grants and revokes).</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>String</td>
<td>Timestamp for the specification of the package body (character data).</td>
</tr>
<tr>
<td>STATUS</td>
<td>String</td>
<td>Status of the package body (VALID, INVALID, or N/A).</td>
</tr>
<tr>
<td>TEMPORARY</td>
<td>String</td>
<td>Whether the package body is temporary (the current session can see only data that it placed in this object itself).</td>
</tr>
<tr>
<td>GENERATED</td>
<td>String</td>
<td>Indicates whether the name of this package body was system generated (Y) or not (N).</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>String</td>
<td>Whether or not this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y</td>
</tr>
</tbody>
</table>
JavaClasses

Table A–19 lists the column name, data type, and description of the JavaClasses Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the Java class.</td>
</tr>
<tr>
<td>NAME</td>
<td>String</td>
<td>Name of the Java class.</td>
</tr>
<tr>
<td>MAJOR</td>
<td>Decimal</td>
<td>Major version number of the Java class, as defined in the JVM specification.</td>
</tr>
<tr>
<td>MINOR</td>
<td>Decimal</td>
<td>Minor version number of the Java class, as defined in the JVM specification.</td>
</tr>
<tr>
<td>KIND</td>
<td>String</td>
<td>Indicates whether the stored object is a Java class (CLASS) or a Java interface (INTERFACE).</td>
</tr>
<tr>
<td>ACCESSIBILITY</td>
<td>String</td>
<td>Accessibility of the Java class.</td>
</tr>
<tr>
<td>IS_INNER</td>
<td>String</td>
<td>Indicates whether this Java class is an inner class (YES) or not (NO).</td>
</tr>
<tr>
<td>IS_ABSTRACT</td>
<td>String</td>
<td>Indicates whether this Java class is an abstract class (YES) or not (NO).</td>
</tr>
<tr>
<td>IS_FINAL</td>
<td>String</td>
<td>Indicates whether this Java class is a final class (YES) or not (NO).</td>
</tr>
<tr>
<td>IS_DEBUG</td>
<td>String</td>
<td>Indicates whether this Java class contains debug information (YES) or not (NO).</td>
</tr>
<tr>
<td>SOURCE</td>
<td>String</td>
<td>Source designation of the Java class.</td>
</tr>
<tr>
<td>SUPER</td>
<td>String</td>
<td>Super class of this Java class.</td>
</tr>
<tr>
<td>OUTER</td>
<td>String</td>
<td>Outer class of this Java class if this Java class is an inner class.</td>
</tr>
</tbody>
</table>

Indexes

Table A–20 lists the column name, data type, and description of the Indexes Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the index.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>String</td>
<td>Name of the index.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>String</td>
<td>Type of the index:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ NORMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ BITMAP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ FUNCTION-BASED NORMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ FUNCTION-BASED BITMAP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ DOMAIN</td>
</tr>
<tr>
<td>TABLE_OWNER</td>
<td>String</td>
<td>Owner of the indexed object.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>String</td>
<td>Name of the indexed object.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>String</td>
<td>Type of the indexed object (for example, TABLE or CLUSTER).</td>
</tr>
<tr>
<td>UNIQUENESS</td>
<td>String</td>
<td>Indicates whether the index is UNIQUE or NONUNIQUE.</td>
</tr>
<tr>
<td>COMPRESSION</td>
<td>String</td>
<td>Indicates whether index compression is enabled (ENABLED) or not (DISABLED).</td>
</tr>
<tr>
<td>PREFIX_LENGTH</td>
<td>Decimal</td>
<td>Number of columns in the prefix of the compression key.</td>
</tr>
<tr>
<td>TABLESPACE_NAME</td>
<td>String</td>
<td>Name of the tablespace containing the index.</td>
</tr>
<tr>
<td>INI_TRANS</td>
<td>Decimal</td>
<td>Initial number of transactions.</td>
</tr>
<tr>
<td>MAX_TRANS</td>
<td>Decimal</td>
<td>Maximum number of transactions.</td>
</tr>
<tr>
<td>INITIAL_EXTENT</td>
<td>Decimal</td>
<td>Size of the initial extent.</td>
</tr>
<tr>
<td>NEXT_EXTENT</td>
<td>Decimal</td>
<td>Size of secondary extents.</td>
</tr>
<tr>
<td>MIN_EXTENTS</td>
<td>Decimal</td>
<td>Minimum number of extents allowed in the segment.</td>
</tr>
<tr>
<td>MAX_EXTENTS</td>
<td>Decimal</td>
<td>Maximum number of extents allowed in the segment.</td>
</tr>
<tr>
<td>PCT_INCREASE</td>
<td>Decimal</td>
<td>Percentage increase in extent size.</td>
</tr>
<tr>
<td>PCT_THRESHOLD</td>
<td>Decimal</td>
<td>Threshold percentage of block space allowed per index entry.</td>
</tr>
<tr>
<td>INCLUDE_COLUMN</td>
<td>Decimal</td>
<td>Column ID of the last column to be included in index-organized table primary key (non-overflow) index. This column maps to the COLUMN_ID column of the *_TAB_COLUMNS data dictionary views.</td>
</tr>
<tr>
<td>FREELISTS</td>
<td>Decimal</td>
<td>Number of process freelists allocated to this segment.</td>
</tr>
<tr>
<td>FREELIST_GROUPS</td>
<td>Decimal</td>
<td>Number of freelist groups allocated to this segment.</td>
</tr>
<tr>
<td>PCT_FREE</td>
<td>Decimal</td>
<td>Minimum percentage of free space in a block.</td>
</tr>
<tr>
<td>LOGGING</td>
<td>String</td>
<td>Logging information.</td>
</tr>
<tr>
<td>BLEVEL</td>
<td>Decimal</td>
<td>B*-Tree level: depth of the index from its root block to its leaf blocks. A depth of 0 indicates that the root block and leaf block are the same.</td>
</tr>
<tr>
<td>LEAF_BLOCKS</td>
<td>Decimal</td>
<td>Number of leaf blocks in the index.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DISTINCT_KEYS</td>
<td>Decimal</td>
<td>Number of distinct indexed values. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is the same as the number of rows in the table (USER_TABLES.NUM_ROWS).</td>
</tr>
<tr>
<td>AVG_LEAF_BLOCKS_PER_KEY</td>
<td>Decimal</td>
<td>Average number of leaf blocks in which each distinct value in the index appears, rounded to the nearest integer. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is always 1.</td>
</tr>
<tr>
<td>AVG_DATA_BLOCKS_PER_KEY</td>
<td>Decimal</td>
<td>Average number of data blocks in the table that are pointed to by a distinct value in the index, rounded to the nearest integer. This statistic is the average number of data blocks that contain rows that contain a given value for the indexed columns.</td>
</tr>
<tr>
<td>CLUSTERING_FACTOR</td>
<td>Decimal</td>
<td>Indicates the amount of order of the rows in the table based on the values of the index.</td>
</tr>
<tr>
<td>STATUS</td>
<td>String</td>
<td>Indicates whether a nonpartitioned index is VALID or UNUSABLE.</td>
</tr>
<tr>
<td>NUM_ROWS</td>
<td>Decimal</td>
<td>Number of rows in the index.</td>
</tr>
<tr>
<td>SAMPLE_SIZE</td>
<td>Decimal</td>
<td>Size of the sample used to analyze the index.</td>
</tr>
<tr>
<td>LAST_ANALYZED</td>
<td>Date</td>
<td>Date on which this index was most recently analyzed.</td>
</tr>
<tr>
<td>DEGREE</td>
<td>String</td>
<td>Number of threads per instance for scanning the index.</td>
</tr>
<tr>
<td>INSTANCES</td>
<td>String</td>
<td>Number of instances across which the indexes to be scanned.</td>
</tr>
<tr>
<td>PARTITIONED</td>
<td>String</td>
<td>Indicates whether the index is partitioned (YES) or not (NO).</td>
</tr>
<tr>
<td>TEMPORARY</td>
<td>String</td>
<td>Indicates whether or not the index is on a temporary table.</td>
</tr>
<tr>
<td>GENERATED</td>
<td>String</td>
<td>Indicates whether the name of the index is system generated (Y) or not (N).</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>String</td>
<td>Indicates whether the index is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y) or not (N).</td>
</tr>
<tr>
<td>BUFFER_POOL</td>
<td>String</td>
<td>Name of the default buffer pool to be used for the index blocks.</td>
</tr>
<tr>
<td>USER_STATS</td>
<td>String</td>
<td>Indicates whether statistics were entered directly by the user (YES) or not (NO).</td>
</tr>
<tr>
<td>DURATION</td>
<td>String</td>
<td>Indicates the duration of a temporary table.</td>
</tr>
<tr>
<td>PCT_DIRECT_ACCESS</td>
<td>Decimal</td>
<td>For a secondary index on an index-organized table, the percentage of rows with VALID guess.</td>
</tr>
<tr>
<td>ITYP_OWNER</td>
<td>String</td>
<td>For a domain index, the owner of the index type.</td>
</tr>
<tr>
<td>ITYP_NAME</td>
<td>String</td>
<td>For a domain index, the name of the index type.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>String</td>
<td>For a domain index, the parameter string.</td>
</tr>
<tr>
<td>GLOBAL_STATS</td>
<td>String</td>
<td>For partitioned indexes, indicates whether statistics are collected by analyzing the index as a whole (YES) or estimated from statistics on underlying index partitions and subpartitions (NO).</td>
</tr>
</tbody>
</table>
IndexColumns

Table A–21 lists the column name, data type, and description of the IndexColumns Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX_OWNER</td>
<td>String</td>
<td>Owner of the index.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>String</td>
<td>Name of the index.</td>
</tr>
<tr>
<td>TABLE_OWNER</td>
<td>String</td>
<td>Owner of the table or cluster.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>String</td>
<td>Name of the table or cluster.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>String</td>
<td>Column name or attribute of object type column.</td>
</tr>
<tr>
<td>COLUMN_POSITION</td>
<td>Decimal</td>
<td>Position of column or attribute within the index.</td>
</tr>
<tr>
<td>COLUMN_LENGTH</td>
<td>Decimal</td>
<td>Indexed length of the column.</td>
</tr>
<tr>
<td>DESCEND</td>
<td>String</td>
<td>Whether the column is sorted in descending order (Y/N).</td>
</tr>
<tr>
<td>CHAR_LENGTH</td>
<td>Decimal</td>
<td>Maximum codepoint length of the column. (Oracle9i or later)</td>
</tr>
</tbody>
</table>

PrimaryKeys

Table A–22 lists the column name, data type, and description of the PrimaryKeys Schema Collection.
### Table A–22 PrimaryKeys

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the constraint definition.</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>String</td>
<td>Name of the constraint definition.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>String</td>
<td>Name associated with the table (or view) with constraint definition.</td>
</tr>
<tr>
<td>SEARCH_CONDITION</td>
<td>String</td>
<td>Text of search condition for a check constraint.</td>
</tr>
<tr>
<td>R_OWNER</td>
<td>String</td>
<td>Owner of table referred to in a referential constraint.</td>
</tr>
<tr>
<td>R_CONSTRAINT_NAME</td>
<td>String</td>
<td>Name of the unique constraint definition for referenced table.</td>
</tr>
<tr>
<td>DELETE_RULE</td>
<td>String</td>
<td>Delete rule for a referential constraint (CASCADE or NO ACTION).</td>
</tr>
<tr>
<td>STATUS</td>
<td>String</td>
<td>Enforcement status of constraint (ENABLED or DISABLED).</td>
</tr>
<tr>
<td>DEFERRABLE</td>
<td>String</td>
<td>Whether or not the constraint is deferrable.</td>
</tr>
<tr>
<td>VALIDATED</td>
<td>String</td>
<td>Whether all data obeys the constraint (VALIDATED or NOT VALIDATED).</td>
</tr>
<tr>
<td>GENERATED</td>
<td>String</td>
<td>Whether the name of the constraint is user or system generated.</td>
</tr>
<tr>
<td>BAD</td>
<td>String</td>
<td>Indicates that this constraint specifies a century in an ambiguous manner. (Yes</td>
</tr>
<tr>
<td>RELY</td>
<td>String</td>
<td>Whether an enabled constraint is enforced or unenforced.</td>
</tr>
<tr>
<td>LAST_CHANGE</td>
<td>DateTime</td>
<td>When the constraint was last enabled or disabled.</td>
</tr>
<tr>
<td>INDEX_OWNER</td>
<td>String</td>
<td>Name of the user owning the index. (Oracle9i or later)</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>String</td>
<td>Name of the index (only shown for unique and primary-key constraints). (Oracle9i or later)</td>
</tr>
</tbody>
</table>

### ForeignKeys

Table A–23 lists the column name, data type, and description of the ForeignKeys Schema Collection.

### Table A–23 ForeignKeys

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY_KEY_</td>
<td>String</td>
<td>Name of the constraint definition.</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td></td>
<td>Owner of the constraint definition.</td>
</tr>
<tr>
<td>PRIMARY_KEY_OWNER</td>
<td>String</td>
<td>Name associated with the table (or view) with constraint definition.</td>
</tr>
<tr>
<td>PRIMARY_KEY_TABLE_NAME</td>
<td>String</td>
<td>Owner of the constraint definition.</td>
</tr>
</tbody>
</table>
### ForeignKeyColumns

Table A–24 lists the column name, data type, and description of the ForeignKeyColumns Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOREIGN_KEY_CONSTRUCTION_NAME</td>
<td>String</td>
<td>Name of the constraint definition.</td>
</tr>
<tr>
<td>FOREIGN_KEY_TABLE_NAME</td>
<td>String</td>
<td>Name associated with the table (or view) with constraint definition.</td>
</tr>
<tr>
<td>SEARCH_CONDITION</td>
<td>String</td>
<td>Text of search condition for a check constraint.</td>
</tr>
<tr>
<td>R_OWNER</td>
<td>String</td>
<td>Owner of table referred to, in a referential constraint.</td>
</tr>
<tr>
<td>RCONSTRAINT_NAME</td>
<td>String</td>
<td>Name of the unique constraint definition for referenced table.</td>
</tr>
<tr>
<td>DELETE_RULE</td>
<td>String</td>
<td>Delete rule for a referential constraint (CASCADE or NO ACTION).</td>
</tr>
<tr>
<td>STATUS</td>
<td>String</td>
<td>Enforcement status of constraint (ENABLED or DISABLED).</td>
</tr>
<tr>
<td>VALIDATED</td>
<td>String</td>
<td>Whether or not all data obeys the constraint (VALIDATED or NOT VALIDATED).</td>
</tr>
<tr>
<td>GENERATED</td>
<td>String</td>
<td>Whether the name of the constraint is user or system generated.</td>
</tr>
<tr>
<td>RELY</td>
<td>String</td>
<td>Whether an enabled constraint is enforced or unenforced.</td>
</tr>
<tr>
<td>LAST_CHANGE</td>
<td>DateTime</td>
<td>When the constraint was last enabled or disabled.</td>
</tr>
<tr>
<td>INDEX_OWNER</td>
<td>String</td>
<td>Name of the user owning the index.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>String</td>
<td>Name of the index.</td>
</tr>
</tbody>
</table>

(Oracle9i or later)

### UniqueKeys

Table A–25 lists the column name, data type, and description of the UniqueKeys Schema Collection.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the constraint definition.</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>String</td>
<td>Name of the constraint definition.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>String</td>
<td>Name of the table with constraint definition.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>String</td>
<td>Name of the column or attribute of the object type column specified in the constraint definition.</td>
</tr>
<tr>
<td>POSITION</td>
<td>String</td>
<td>Original position of column or attribute in the definition of the object.</td>
</tr>
</tbody>
</table>
### Table A–25  UniqueKeys

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>String</td>
<td>Owner of the constraint definition.</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>String</td>
<td>Name of the constraint definition.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>String</td>
<td>Name associated with the table (or view) with constraint definition.</td>
</tr>
<tr>
<td>SEARCH_CONDITION</td>
<td>String</td>
<td>Text of search condition for a check constraint.</td>
</tr>
<tr>
<td>R_OWNER</td>
<td>String</td>
<td>Owner of table referred to in a referential constraint.</td>
</tr>
<tr>
<td>R_CONSTRAINT_NAME</td>
<td>String</td>
<td>Name of the unique constraint definition for referenced table.</td>
</tr>
<tr>
<td>DELETE_RULE</td>
<td>String</td>
<td>Delete rule for a referential constraint (CASCADE or NO ACTION).</td>
</tr>
<tr>
<td>STATUS</td>
<td>String</td>
<td>Enforcement status of constraint (ENABLED or DISABLED).</td>
</tr>
<tr>
<td>DEFERRABLE</td>
<td>String</td>
<td>Whether or not the constraint is deferrable.</td>
</tr>
<tr>
<td>VALIDATED</td>
<td>String</td>
<td>Whether all data obeys the constraint (VALIDATED or NOT VALIDATED).</td>
</tr>
<tr>
<td>GENERATED</td>
<td>String</td>
<td>Whether the name of the constraint is user or system generated.</td>
</tr>
<tr>
<td>BAD</td>
<td>String</td>
<td>Indicates that this constraint specifies a century in an ambiguous manner.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>using the TO_DATE function with a four-digit year.</td>
</tr>
<tr>
<td>RELY</td>
<td>String</td>
<td>Whether an enabled constraint is enforced or not.</td>
</tr>
<tr>
<td>LAST_CHANGE</td>
<td>String</td>
<td>When the constraint was last enabled or disabled.</td>
</tr>
<tr>
<td>INDEX_OWNER</td>
<td>String</td>
<td>Name of the user owning the index.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Oracle9i or later)</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>String</td>
<td>Name of the index (only shown for unique and primary-key constraints).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Oracle9i or later)</td>
</tr>
</tbody>
</table>
This appendix lists the Entity Framework canonical functions and the corresponding ODP.NET provider functions to which they map.

**Aggregate Canonical Functions**

<table>
<thead>
<tr>
<th>Canonical Function</th>
<th>Oracle Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg (expression)</td>
<td>AVG(expression)</td>
</tr>
<tr>
<td>BigCount ( expression)</td>
<td>COUNT(expression)</td>
</tr>
<tr>
<td>Count ( expression )</td>
<td>COUNT(expression)</td>
</tr>
<tr>
<td>Max ( expression )</td>
<td>MAX(expression)</td>
</tr>
<tr>
<td>Min ( expression )</td>
<td>MIN(expression)</td>
</tr>
<tr>
<td>StDev ( expression )</td>
<td>STDDEV(expression)</td>
</tr>
<tr>
<td>StDevP ( expression )</td>
<td>STDEVP(expression)</td>
</tr>
<tr>
<td>Sum ( expression )</td>
<td>SUM(expression)</td>
</tr>
<tr>
<td>Var(expression)</td>
<td>VAR(expression)</td>
</tr>
<tr>
<td>VarP(expression)</td>
<td>VARP(expression)</td>
</tr>
</tbody>
</table>

**Math Canonical Functions**

<table>
<thead>
<tr>
<th>Canonical Function</th>
<th>Oracle Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs (value)</td>
<td>ABS (value)</td>
</tr>
<tr>
<td>Ceiling (value)</td>
<td>CEIL(value)</td>
</tr>
<tr>
<td>Floor (value)</td>
<td>FLOOR(value)</td>
</tr>
<tr>
<td>Power(value, exponent)</td>
<td>POWER(value, exponent)</td>
</tr>
<tr>
<td>Round (value)</td>
<td>ROUND(value)</td>
</tr>
<tr>
<td>Round (value, digits)</td>
<td>ROUND(value, digits)</td>
</tr>
<tr>
<td>Truncate(value, digits)</td>
<td>TRUNC(value, digits)</td>
</tr>
</tbody>
</table>
String Canonical Functions

<table>
<thead>
<tr>
<th>Canonical Function</th>
<th>Oracle Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concat (string1, string2)</td>
<td>CONCAT(string1, string2) or ( (string1)</td>
</tr>
<tr>
<td>Contains(string, target)</td>
<td>INSTR(string, target)</td>
</tr>
<tr>
<td>EndsWith(string, target)</td>
<td>INSTR(REVERSE(string), REVERSE(target))</td>
</tr>
<tr>
<td>Comparison operators</td>
<td>Comparison operators</td>
</tr>
<tr>
<td>(&lt;, &lt;=, &gt;, &gt;=, &lt;&gt;, !&gt;)</td>
<td>(&lt;, &lt;=, &gt;, &gt;=, &lt;&gt;, !&gt;)</td>
</tr>
<tr>
<td>IndexOf(target, string)</td>
<td>INSTR(string2, target)</td>
</tr>
<tr>
<td>Left (string1, length)</td>
<td>SUBSTR(string1, length)</td>
</tr>
<tr>
<td>Length (string)</td>
<td>LENGTH(string)</td>
</tr>
<tr>
<td>LTrim(string)</td>
<td>LTRIM(string)</td>
</tr>
<tr>
<td>Replace (string1, string2, string3)</td>
<td>REPLACE(string1, string2, string3)</td>
</tr>
<tr>
<td>Reverse (string)</td>
<td>REVERSE(string)</td>
</tr>
<tr>
<td>Right (string, length)</td>
<td>(CASE WHEN LENGTH(string) &gt;= (length) THEN SUBSTR(string), -(length), length) ELSE string END)</td>
</tr>
<tr>
<td>RTrim(string)</td>
<td>RTRIM(string)</td>
</tr>
<tr>
<td>Substring (string, start, length)</td>
<td>SUBSTR(( string, start, length)</td>
</tr>
<tr>
<td>StartsWith(string, target)</td>
<td>INSTR(string, target)</td>
</tr>
<tr>
<td>ToLower (string)</td>
<td>LOWER(string)</td>
</tr>
<tr>
<td>ToUpper (string)</td>
<td>UPPER</td>
</tr>
<tr>
<td>Trim (string)</td>
<td>LTRIM(RTRIM(string))</td>
</tr>
</tbody>
</table>

Date And Time Canonical Functions

<table>
<thead>
<tr>
<th>Canonical Function</th>
<th>Oracle Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddNanoseconds(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>AddMicroseconds(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>AddMilliseconds(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>AddSeconds(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>AddMinutes(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>AddHours(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>AddDays(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>Canonical Function</td>
<td>Oracle Function</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>AddMonths(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>AddYears(expression, number)</td>
<td>(expression) + INTERVAL</td>
</tr>
<tr>
<td>CreateDateTime(year, month, day, hour, minute, second)</td>
<td>TO_TIMESTAMP</td>
</tr>
<tr>
<td>CreateDateTimeOffset(year, month, day, hour, minute, second, tzoffset)</td>
<td>TO_TIMESTAMP_TZ</td>
</tr>
<tr>
<td>CreateTime(hour, minute, second)</td>
<td>Time literals are not supported in Oracle</td>
</tr>
<tr>
<td>CurrentDateTime()</td>
<td>LOCALTIMESTAMP</td>
</tr>
<tr>
<td>CurrentDateTimeOffset()</td>
<td>SYSTIMESTAMP</td>
</tr>
<tr>
<td>CurrentUtcDateTime()</td>
<td>SYS_EXTRACT_UTC (LOCALTIMESTAMP)</td>
</tr>
<tr>
<td>Day(expression)</td>
<td>EXTRACT(DAY FROM expression)</td>
</tr>
<tr>
<td>DayOfYear(expression)</td>
<td>TO_NUMBER(TO_CHAR(CAST(expression AS TIMESTAMP), 'DDD'))</td>
</tr>
<tr>
<td>DiffNanoseconds(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>DiffMilliseconds(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>DiffMicroseconds(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>DiffSeconds(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>DiffMinutes(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>DiffHours(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>DiffDays(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>DiffMonths(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>DiffYears(startExpression, endExpression)</td>
<td>EXTRACT and arithmetic operations</td>
</tr>
<tr>
<td>Comparison operators (&lt;, &lt;=, &gt;, &gt;=, &lt;&gt;, !=)</td>
<td>&lt;, &lt;=, &gt;, &gt;=, &lt;&gt;, != operators</td>
</tr>
<tr>
<td>GetTotalOffsetMinutes (datetimeoffset)</td>
<td>(EXTRACT(TIMEZONE_HOUR FROM (expression))) * 60 + EXTRACT (TIMEZONE_MINUTE FROM(expression))</td>
</tr>
<tr>
<td></td>
<td>(Require multiple operations.)</td>
</tr>
<tr>
<td>Hour (expression)</td>
<td>EXTRACT(HOUR FROM expression)</td>
</tr>
</tbody>
</table>
### Canonical Function Oracle Function

<table>
<thead>
<tr>
<th>Canonical Function</th>
<th>Oracle Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Millisecond(expression)</td>
<td>NVL(TO_NUMBER(SUBSTR(CAST(expression AS TIMESTAMP), 'DD-MON-RR HH24:MI:SSXFF'), 20, 3)), 0)</td>
</tr>
<tr>
<td>Minute(expression)</td>
<td>EXTRACT(MINUTE FROM expression)</td>
</tr>
<tr>
<td>Month (expression)</td>
<td>EXTRACT(MONTH FROM expression)</td>
</tr>
<tr>
<td>Second(expression)</td>
<td>EXTRACT(SECOND FROM expression)</td>
</tr>
<tr>
<td>TruncateDate(expression)</td>
<td>TRUNC(expression)</td>
</tr>
<tr>
<td>Year(expression)</td>
<td>EXTRACT(YEAR FROM expression)</td>
</tr>
</tbody>
</table>

### Bitwise Canonical Functions

<table>
<thead>
<tr>
<th>Canonical Function</th>
<th>Oracle Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitWiseAnd (value1, value2)</td>
<td>BITAND(value1, value2)</td>
</tr>
<tr>
<td>BitWiseNot (value)</td>
<td>( 0 - value ) - 1</td>
</tr>
<tr>
<td>BitWiseOr (value1, value2)</td>
<td>Value1 - BITAND(value1, value2) + value2</td>
</tr>
<tr>
<td>BitWiseXor (value1, value2)</td>
<td>Value1 - 2 * BITAND(value1, value2) + value2</td>
</tr>
</tbody>
</table>

### Other Canonical Functions

<table>
<thead>
<tr>
<th>Canonical Function</th>
<th>Oracle Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewGuid()</td>
<td>SYS_GUID</td>
</tr>
</tbody>
</table>
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 data type 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 data type 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.

DataSet
A DataSet is an in-memory copy of database data. The DataSet exists in memory without an active connection to the database.

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.
Extensible Stylesheet Language Transformation (XSLT)
The XSL W3C standard specification that defines a transformation language to convert one XML document into another.

flush
Flush or flushing refers to recording changes (that is, sending modified data) to the database.

Global Assembly Cache (GAC)
A cache for .NET assemblies.

goodness
The degree of load in the Oracle database. The lighter load is better and vice versa.

implicit database connection
The connection that is implicitly available from the context of the .NET stored procedure execution.

instantiate
A term used in object-based languages such as C# to refer to the creation of an object of a specific class.

invalidation message
The content of a change notification which indicates that the cache is now invalid

Large Object (LOB)
The class of SQL data type 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 data type 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 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**
The Oracle 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*.

**REF**
A data type that encapsulates references to row objects of a specified object type.

**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.
A

abstract data types, 3-99
ADO, 1-3
ADO.NET, 1-3, 3-113
ADO.NET 1.x, 2-1
ADO.NET 2.0, 2-3, 3-20
  base or DbCommon classes, 3-20
  classes and class members, 3-29
DbCommon classes, 3-20
ADO.NET 2.x, 2-1
ADTs, 3-99
Advanced Queuing support, xxvi
app.config
  sample, 2-13
  setting performance counters, 3-9
app.config file, 3-21
AppDomain, 3-9
application config, 2-5
AQ support, xxvi
array bind
  OracleParameter, 3-54
array bind operations, 3-53
  ArrayBindCount, 5-13
  ArrayBindIndex, 5-220
  ArrayBindSize, 5-265
  ArrayBindStatus, 5-266
  error handling, 3-55
array binding, 3-53
ArrayBindCount property, 5-13
ArrayBindIndex property, 5-220
ArrayBindSize property, 3-50, 3-54, 5-265
ArrayBindStatus property, 3-50, 3-54, 5-266
ASP.NET, 1-2
assembly, 1-4
  ODP.NET, 1-4

B

batch processing, 3-29
  support, 3-20
BatchUpdate
  Microsoft Hotfix, 3-29
behavior of ExecuteScalar method for REF CURSOR, 3-74
BFILE, 3-83
BINARY_DOUBLE, 3-45
BINARY_FLOAT, 3-45
binding, 3-45
  PL/SQL Associative Array, 3-50
BLOB, 3-83
bulk copy constraints, 3-30
bulk copy feature, 3-30
  restrictions, 3-30

C

C#, Visual Basic .NET, C++ .NET, 1-1
caching ODP.NET parameter contexts, xxix
callback support, 3-17
case-sensitivity
  column name mapping, 3-97
change notification
  ODP.NET support, 3-122
change notification, Database Change
  Notification, 9-1
characters with special meaning
  in column data, 3-96
  in table or view, 3-97
characters with special meaning in XML, 3-90
client applications, 1-1
client globalization settings, 3-133, 3-135
client identifier, 3-17
CLOB, 3-83
close calls, 2-14
CLR, 1-2
collection types, 3-100
CollectionType property, 3-50
column data
  special characters in, 3-96
CommandBehavior.SequentialAccess, 3-65
commit transactions
  changes to XML data, 3-99
CommittableTransaction, 3-25
config files
  samples, 2-13
configuration settings
  UDTs, 3-114
connect descriptor, 3-4
connection dependency, 3-88
connection optimization, 3-10
connection pool
performance counters, 3-7
connection pooling, 3-5
example, 3-5
for Oracle RAC database, 3-10
management, 3-7
Connection property, 3-85
connection string builder, 3-20
ConnectionString attributes, 3-5
Connection Lifetime, 3-2, 3-5, 3-6
Connection Timeout, 3-2, 3-5, 3-6
Data Source, 3-2
DBA Privilege, 3-2
Decr Pool Size, 3-2, 3-5, 3-6
Enlist, 3-2
HA Events, 3-2, 3-5
Incr Pool Size, 3-2, 3-5, 3-6
Load Balancing, 3-2, 3-5
Max Pool Size, 3-2, 3-5, 3-6
Metadata Pooling, 3-2
Min Pool Size, 3-2, 3-5, 3-6
Password, 3-2
Persist Security Info, 3-2
Pooling, 3-2, 3-5, 3-6
Proxy Password, 3-2, 3-15
Proxy User Id, 3-2, 3-15
Statement Cache Purge, 3-2
Statement Cache Size, 3-2
User Id, 3-2
Validate Connection, 3-2, 3-5, 3-6
ConnectionString property, 3-5, 3-6, 5-78
Constraints property, 3-131
configuring, 3-132
context connection, 4-2
Continuous Query Notification, 3-120
controlling query reexecution, 3-130
custom classes, 3-100
custom type factories, 3-100
custom type factory, 3-103
custom type implementations
optional, 3-102
custom type mapping, 3-103
custom type mappings
specifying, 3-103
specifying with custom type factories, 3-103
specifying with XML, 3-103
using, 3-105
custom types
converting to Oracle UDTS, 3-105
requirements, 3-101
custom UDT classes, 3-100

D
data loss, 3-127
data manipulation
using XML, 3-95
data source attribute, 3-4
data source enumerator, 3-20
data source enumerators, 3-21
database
changes to, 3-95
Database Change Notification
best practices, 3-126
performance considerations, 3-126
database change notification, 2-13, 3-120, 3-122
ODP.NET support, 3-122
database notification
port to listen, 2-13
database shutdown, xxvii
database startup, xxvii
DataSet, 3-85
populating, 3-73
populating from a REF CURSOR, 3-73
populating with generic and custom
objects, 3-113
updating, 3-74
updating to database, 3-131
DataSet Fill, xxvi
DataTable, 3-132
Datatable properties, 3-131
DbCommon classes, 3-20
DBlinks, 4-2
DbProviderFactories class, 2-3, 3-20
DbType
inference, 3-47
debug tracing, 3-136
default mapping
improving, 3-99
dependent unmanaged DLL mismatch, 2-5
direct path load, 3-30
dispose calls, 2-14
distributed transactions, 3-25
documentation
.NET, 1-1
Dynamic Enlistment, 3-16
dynamic help, 1-1, 2-3

E
Easy Connect naming method, 3-4
EDM type facets, 3-37
EDM types
and Oracle data types, 3-33
direct-to-end tracing, xxviii, 3-17
EnlistDistributedTransaction method, 3-16
Entity Framework, xxiii, 3-32
enumeration type
OracleDbType, 3-46
error handling, 3-55
example
custom type pooling, 3-5
datatet examples
documentation
readme file, 2-4
ExecuteNonQuery method, 3-73
ExecuteScalar method, 3-74
explicit user connections, 4-1
EZCONNECT, 3-4
failover, 3-17
  registering an event handler, 3-17
FailoverEvent Enumeration
description, 11-10
FailoverReturnCode Enumeration
description, 11-11
FailoverType Enumeration
description, 11-12
FAN, 3-10
Fast Application Notification (FAN), 3-10
features, 3-1
  new, xxiii
FetchSize property
  fine-tuning, 3-71
  setting at design time, 3-71
  setting at run time, 3-72
  using, 3-70
file locations, 2-4

garbage collection, 2-14
geographic data, 3-99
Global Assembly Cache (GAC), 2-3
globalization settings, 3-133
  client, 3-133
  session, 3-134
  thread-based, 3-134
globalization support, 3-133
globalization-sensitive operations, 3-135
Grid environment, 3-10
grid-computing, xxxi, xxxii
grids, xxxi, xxxii
GUI access to ODP.NET, 1-2

HA Events, 3-2, 3-10, 3-11
HA events, 2-13
handling date and time format
  manipulating data in XML, 3-95
  retrieving queries in XML, 3-91

implicit database connection, 4-1, 4-2, 4-3, 5-84
implicit REF CURSOR, xxiv, 3-75
  bind and metadata, 3-76
  bindinfo, 3-78
  configuration, 3-79
  examples, 3-79
  usage, 3-82
improving default mapping, 3-99
inference from Value property, 3-49
inference of DbType and OracleDbType from Value, 3-49
inference of DbType from OracleDbType, 3-47
inference of OracleDbType from DbType, 3-48
inference of types, 3-47

InitialLOBFetchSize property, 3-66
InitialLONGFetchSize property, 3-65
input binding
  XMLType column, 3-89
installation, 2-3
  Oracle Data Provider for .NET, 2-3
  XCopy class, 2-3
integrated help, 2-3
interference in OracleParameter class, 3-47
introduction, overview, 1-3
INullable Interface
  interface description, 14-409
  interface members, 14-410
  interface properties, 14-411
invalidation message, 3-122
  ensuring persistency of, 3-122
InvalidCastException, 3-62
IOraclArrayTypeFactory Interface
  interface description, 16-33
  interface members, 16-34
  interface methods, 16-35
IOraclCustomType Interface
  interface description, 16-26
  interface members, 16-27
  interface methods, 16-28
IOraclCustomTypeFactory Interface
  interface description, 16-30
  interface members, 16-31
  interface methods, 16-32

large binary datatypes, 3-84
large character datatypes, 3-84
limitations and restrictions, 4-2
LINQ to Entities, xxiii, 3-32
Load Balancing, 3-2, 3-10
load balancing, 2-13
LOB Connection property, 3-85
LOB retrieval, xxix
LOBs
  temporary, 3-86
  updating, 3-85, 3-86
LOBs updating, 3-85
local transactions, 3-25
location data, 3-99
LONG and LONG RAW datatypes, 3-84

machine.config, 2-5
machine.config file, 2-3
metadata, 3-132
method invocation
  UDT, 3-114
Microsoft ADO.NET 2.0, 3-20
Microsoft Common Language Runtime (CLR), 1-2
Microsoft Hotfix
  BatchUpdate, 3-29
Microsoft .NET Framework, 2-1
Microsoft .NET Framework Class Library, 1-3
Microsoft Transaction Server, 2-2
MTS, 2-2
multiple notification requests, 3-122
Multiple Oracle Homes, xxxiii
multiple tables
changes to, 3-99

N
namespace
   Oracle.DataAccess.Types, 1-9
native XML support, 3-86
NCLOB, 3-83
nested table types, 3-100
.NET custom types, 3-99
.NET Framework datatype, 3-60
.NET languages, 1-1, 1-2
.NET products and documentation, 1-1
.NET stored procedures and functions, 4-1
.NET Stream class, 3-84
.NET type accessors, 3-62
.NET Types
   inference, 3-47
notification framework, 3-122
notification information
   retrieving, 3-122
notification process
   flow, 3-124
notification registration, 3-122
   requirements of, 3-124
NULL values
   retrieving from column, 3-95
   number of rows fetched in round-trip
   controlling, 3-70

O
object data type support, 3-99
object types, 3-100
object-relational data, 3-94
   saving changes from XML data, 3-99
obtaining a REF CURSOR, 3-72, 3-73
obtaining an OracleRefCursor, 3-72
obtaining data from an OracleDataReader, 3-62
obtaining LOB data
   InitialLOBFetchSize property, 3-66
obtaining LONG and LONG RAW Data, 3-65
OCI
   statement caching, 3-56
OData, xxiv
ODP.NET
   installing, 2-3
   ODP.NET Configuration, 2-5
   ODP.NET LOB classes, 3-83
   ODP.NET Type accessors, 3-64
   ODP.NET Type classes, 3-60
   ODP.NET Type exceptions, 15-1
   ODP.NET Type structures, 3-60, 14-1
   ODP.NET Types, 3-60
   overview, 3-60
ODP.NET within a .NET stored procedure
   limitations and restrictions, 4-2
   transaction support, 4-3
   unsupported SQL commands, 4-6
ODP.NET XML Support, 3-86
OnChangedEventArgs Class
   instance properties, 9-34
   members, 9-31
   static fields, 9-32
   static methods, 9-33
OnChangeEventHanlder Delegate
   description, 9-40
operating system authentication, 3-12
Oracle Call Interface
   statement caching, 3-56
Oracle Data Guard
   connecting, 3-10
Oracle Data Provider for .NET
   installing, 2-3
   system requirements, 2-1
Oracle Data Provider for .NET assembly, 1-4
Oracle data types
   and EDM types, 3-33
   mapping and customizing, 3-42
Oracle Database Extensions for .NET, 1-2, 4-1
Oracle Developer Tools for Visual Studio, 1-2
Oracle Label Security, 3-17
Oracle native types, 3-60
   supported by ODP.NET, 3-62
Oracle Providers for ASP.NET, 1-2
Oracle RAC
   connecting, 3-10
Oracle RAC database
   pool size attributes, 3-12
Oracle RAC environment, 3-10
Oracle Services for Microsoft Transaction Server, 2-2
Oracle UDT attribute mappings, 3-106
Oracle Universal Installer (OUI), 2-3
Oracle user-defined types, 3-99
Oracle User-Defined Types (UDTs), 3-100
Oracle Virtual Private Database (VPD), 3-17
Oracle XML DB, 3-87
Oracle8i Database, ADO.NET 2.0
   interfaces, 3-20
OracleAQAgent
   constructors, 12-4
OracleAQAgent Class
   constructors, 12-4
   description, 12-2
   members, 12-3, 12-39
   properties, 12-6, 12-12
OracleAQDequeueMode Enumeration
   description, 12-84
OracleAQDequeueOptions Class
   constructor, 12-11
   description, 12-8
   members, 12-9
   properties, 12-12
   public methods, 12-18
OracleAQEnqueueOptions Class
constructor, 12-21
description, 12-19
members, 12-20
properties, 12-22
public methods, 12-24
OracleAQMessage Class
constructors, 12-28
description, 12-25
members, 12-26
properties, 12-30
OracleAQMessageAvailableEventArgs Class
description, 12-38
members, 12-39
properties, 12-40
OracleAQMessageAvailableEventHandler Delegate
description, 12-47
OracleAQMessageDeliveryMode Enumeration
description, 12-85
OracleAQMessageState Enumeration
description, 12-87
OracleAQMessageType Enumeration
description, 12-88
OracleAQNavigationMode Enumeration
description, 12-89
OracleAQNotificationGroupingType Enumeration
description, 12-90
OracleAQNotificationType Enumeration
description, 12-91
OracleAQQueue Class
constructors, 12-51
description, 12-48
events, 12-79
members, 12-49
properties, 12-58
public methods, 12-64
static methods, 12-51
OracleAQVisibilityMode Enumeration
description, 12-92
OracleArrayMappingAttribute Class
constructors, 16-22
description, 16-19
members, 16-20
methods, 16-25
properties, 16-24
static methods, 16-23
OracleBFile Class
class description, 13-2
constructors, 13-7
instance methods, 13-19
instance properties, 13-12
members, 13-4
static fields, 13-9
static methods, 13-11
OracleBinary Structure
constructor, 14-7
description, 14-2
instance methods, 14-26
members, 14-4
properties, 14-23
static fields, 14-8
static methods, 14-9
static operators, 14-15
static type conversion operators, 14-21
OracleBlob Class
class description, 13-38
constructors, 13-43
instance methods, 13-53
instance properties, 13-47
members, 13-40
static fields, 13-45
static methods, 13-46
OracleBulkCopy Class
class description, 17-2
constructors, 17-5
events, 17-20
members, 17-3
properties, 17-9
public methods, 17-14
OracleBulkCopyColumnMapping Class
class description, 17-21
constructors, 17-23
members, 17-22
properties, 17-26
OracleBulkCopyColumnMappingCollection Class
class description, 17-29
members, 17-31
properties, 17-32
public methods, 17-33
OracleBulkCopyOptions Enumeration
description, 17-42
OracleClientFactory, 2-3
OracleClientFactory Class
class description, 7-2
class members, 7-4
public methods, 7-6
public properties, 7-5
OracleClientFactory class
instantiating, 3-20
OracleClob Class
class description, 13-73
constructors, 13-78
instance methods, 13-88
instance properties, 13-82
members, 13-75
static fields, 13-80
static methods, 13-81
OracleCollectionType Enumeration, 5-357
OracleCommand
ArrayBindCount property, 3-53
constructors, 5-7
InitialLOBFetchSize property, 3-66
InitialLONGFetchSize property, 3-65
Transaction property, 3-44
OracleCommand Class
ArrayBindCount, 5-13
class description, 5-2
ExecuteScalar method, 3-74
FetchSize property, 3-70
members, 5-4
properties, 5-10
public methods, 5-27, 12-18
RowSize property, 3-71
static methods, 5-9
OracleCommand object, 3-44
OracleCommand properties
ArrayBindCount, 3-53
OracleCommand Transaction object, 3-44
OracleCommandBuilder Class, 3-132
class description, 5-41
constructors, 5-47
events, 5-63
members, 5-44
properties, 5-53
public methods, 5-58
static methods, 5-49
updating dataset, 3-131
OracleConnection
ClearAllPools property, 3-7
ClearPool property, 3-7
ClientId property, 3-17
events, 5-112
OracleConnection Class
class description, 5-64
constructors, 5-69
members, 5-66
obtaining a reference, 3-88
properties, 5-76
public methods, 5-90
static methods, 5-73
OracleConnection class
GetSchema methods, 3-23
OracleConnectionStringBuilder Class
class description, 7-10
class members, 7-13
constructors, 7-16
public methods, 7-33
public properties, 7-18
OracleConnectionStringBuilder class using, 3-21
OracleConnectionType Enumeration
description, 5-356
OracleCustomTypeMappingAttribute Class
constructors, 16-6
description, 16-2
members, 16-4
methods, 16-9
properties, 16-8
static methods, 16-7
Oracle.DataAccess.Client namespace, 1-4
Oracle.DataAccess.dll, 1-4
Oracle.DataAccess.Types namespace, 1-4, 1-9
OracleDataAdapter, 3-127
constructors, 5-121
members, 5-118
SafeMapping Property, 3-128
SelectCommand property, 3-73
OracleDataAdapter Class, 5-116
events, 5-136
FillSchema method, 3-132
properties, 5-125
public methods, 5-131
SelectCommand property, 3-132
static methods, 5-124
OracleDataAdapter class
FillSchema method, 3-132
Requery property, 3-130
SelectCommand property, 3-131
OracleDataAdapter Safe Type Mapping, 3-127
OracleDatabase Class
class description, 5-140
constructor, 5-143
members, 5-142
properties, 5-144
public methods, 5-145
Shutdown method, 5-145
Startup method, 5-149
OracleDataReader, 3-62, 3-65
members, 5-155
retrieving UDTs from, 3-108
typed accessors, 3-62
OracleDataReader Class
class description, 5-152
FetchSize property, 3-70
populating, 3-73
properties, 5-160
public methods, 5-171
static methods, 5-159
OracleDataReader Class SchemaTable, 5-206
OracleDataSource Enumerator class using, 3-21
OracleDataSourceEnumerator Class
class description, 7-36
class members, 7-38
public methods, 7-39
OracleDate Structure
constructors, 14-34
description, 14-29
members, 14-31
methods, 14-60
properties, 14-56
static fields, 14-39
static methods, 14-41
static operators, 14-47
static type conversions, 14-52
OracleDBShutdownMode Enumeration
description, 5-358
OracleDBStartupMode Enumeration
description, 5-359
OracleDbType
inference, 3-47
OracleDbType enumeration, 3-47
OracleDbType enumeration type, 3-46, 5-360
OracleDecimal Structure
constructors, 14-72
description, 14-65
instance methods, 14-133
members, 14-67
properties, 14-129
static comparison methods, 14-82
Index-6
static comparison operators, 14-112
static logarithmic methods, 14-101
static manipulation methods, 14-87
static operators, .NET Type to OracleDecimal, 14-120
static operators, OracleDecimal to .NET, 14-124
static trigonometric methods, 14-106

OracleDependency Class
change notification, 3-122
class description, 9-2
constructors, 9-5
database change notification, 3-120
events, 9-21
instance methods, 9-18
instance properties, 9-12
members, 9-3
static fields, 9-9
static methods, 9-11

OracleError Class
ArrayBindIndex, 5-220
class description, 5-216
members, 5-218
methods, 5-223
properties, 5-220
static methods, 5-219

OracleErrorCollection
members, 5-226
properties, 5-228
public methods, 5-229
static methods, 5-227

OracleErrorCollection Class, 5-224

OracleException Class, 5-230

OracleFailoverEventArgs
members, 11-5
properties, 11-7
public methods, 11-8

OracleFailoverEventHandler Delegate
description, 11-9

OracleGlobalization Class
class description, 10-2
members, 10-4
properties, 10-12
public methods, 10-22

OracleHAEventArgs Class
description, 8-2
members, 8-3
properties, 8-4

OracleHAEventHandler Delegate
description, 8-8

OracleHAEventSource Enumeration
description, 8-9

OracleHAEventStatus Enumeration
description, 8-10

OracleInfoMessageEventArgs
members, 5-242

OracleInfoMessageEventHandler Delegate, 5-247

OracleIntervalDS Structure
constructors, 14-144
description, 14-139
members, 14-141
methods, 14-174
properties, 14-169
static methods, 14-151
static operators, 14-158
type conversions, 14-166

OracleIntervalYM Structure
constructors, 14-182
description, 14-177
members, 14-179
methods, 14-188, 14-207
properties, 14-204
static fields, 14-186
static operators, 14-194
type conversions, 14-201

OracleNotificationEventArgs
change notification, 3-122
class description, 9-30
instance methods, 9-39

OracleNotificationInfo Enumeration
description, 9-44

OracleNotificationRequest Class
database change notification, 3-120
instance methods, 9-29
instance properties, 9-25
members, 9-23
static methods, 9-24

OracleNotificationSource Enumeration
description, 9-43

OracleNotificationType Enumeration
description, 9-42

OracleNullValueException Class
class description, 15-11
constructors, 15-14
members, 15-12
methods, 15-16, 15-18
properties, 15-17

OracleObjectMappingAttribute Class
constructors, 16-13
description, 16-10
members, 16-11
methods, 16-18
properties, 16-16
static methods, 16-15

OracleParameter
array bind properties, 3-54
ArrayBindSize property, 3-54, 5-265
ArrayBindStatus property, 3-54, 5-266
constructors, 5-252
inferences of types, 3-47
members, 5-250
parameter binding, 3-45
parameter binding with OracleParameter, 3-110
parameter context caching, xxix
password expiration, 3-14
passwords in code examples, xxi
performance, 3-56
  array binding, 3-53
  connection pooling, 3-5
  fine-tuning FetchSize, 3-71

number of rows fetched, 3-70
Obtaining LOB Data, 3-66
performance counters
  connection pool, 3-7
  instance names of, 3-9
  publishing, 3-7
  using app.config entry, 3-9
PL/SQL Associative Array binding, 3-50
PL/SQL Index-By Tables, 3-50
PL/SQL language, 3-72
PL/SQL REF CURSOR, 3-72
PL/SQL REF CURSOR and OracleRefCursor, 3-72
PLSQLAssociativeArray, 5-357
populating an OracleDataReader from a REF CURSOR, 3-73
populating an OracleRefCursor from a REF CURSOR, 3-73
populating the DataSet from a REF CURSOR, 3-73
populating the DataSet with generic and custom objects, 3-113
port
  listen for database notifications, 2-13
port number
  defining listener, 3-122
porting
  client application to .NET stored procedure, 4-6
  preventing data loss, 3-127, 3-128
  PrimaryKey property, 3-131
  configuring, 3-132
  privileged connections, 3-13
  promotable transactions, xxvi, 3-25
properties
  ClientId property, 3-17
  provider factory classes, 3-20
  provider independence, 3-20
  proxy authentication, 3-15

Q
query result set
  retrieving as XML, 3-91

R

REF CURSOR
  behavior of ExecuteScalar method, 3-74
  implicit, xxiv, 3-75
  obtaining, 3-72, 3-73
  passing to stored procedure, 3-74
  populating DataSet from, 3-73
  populating from OracleDataReader, 3-73
registry entries, 2-5
release Oracle8i (8.1.7), 3-95
release Oracle9i(9.0.x), 3-95
Requery property, 3-130
round-trip, 3-53
RowSize property, 3-71
Runtime Connection Load Balancing, 3-10
S

Safe Type Mapping, 3-127
SafeMapping Property, 3-128
Samples, 1-12
samples, 2-4
UDT, 16-1
saving change using an XML document, 3-98
saving changes
using XML data, 3-96
schema metadata
customizing metadata, 3-23
SchemaTable, 5-206
search order
unmanaged DLLs, 2-4
SecureFiles, xxix, 3-83
security enhancements, xxvi
SelectCommand property, 3-73
Self-Tuning, xxv
session globalization parameters, 3-136
session globalization settings, 3-134
schema discovery
support, 3-20
Shutdown method, 5-145
shutdown, database, xxvii
simple application, 1-12
Size property, 3-50
SQL commands
unsupported, 4-6
Startup method, 5-149
startup, database, xxvii
Statement Caching
connection string attributes, 3-57
methods and properties, 3-57
Statement Cache Purge, 3-56
Statement Cache Size, 3-56
StatementCacheWithUdts, 3-114
stored procedures and functions, 3-74, 4-1
Stream class, 3-84
support comparison
client application versus .NET stored
procedure, 4-6
SYSDBA privileges, 3-13
SYSOPER privileges, 3-13
system requirements
Oracle Data Provider for .NET, 2-1
System.Data.Common, 3-20
System.Transactions support, 3-25

T

table or view
special characters in, 3-97
TAF, 3-17
TAF callback support, 3-17
Temporary LOBs, 3-86
thread globalization settings, 3-136
Thread.Abort method, 2-14
thread-based globalization settings, 3-134
tips for ODP.NET, 2-14
TNS alias, 3-4

tracing
end-to-end, xxviii
tracing attributes, 3-17
Transaction object, 3-44
Transaction property, 3-44
transaction support, 4-3
transactions
commit, 3-99
TransactionScope, 3-25
Transparent Application Failover (TAF), 3-17
troubleshooting, 3-136
typed OracleDataReader accessors, 3-62

U

UDT
method invocation, 3-114
UDT metadata retrieval from
OracleDataReader, 3-109
UdtCacheSize, 3-114
UDTs, 3-99
collection types, 3-100
collection settings, 3-114
object types, 3-100
parameter binding with OracleParameter, 3-110
retrieving from OracleDataReader, 3-108
samples, 16-1
UDTs (Oracle User-Defined Types), 3-100
UdtTypeName property, 3-110
unique columns, 3-65, 3-66
unique constraint, 3-65, 3-66
unique index, 3-65, 3-66
UniqueConstraint, 3-132
uniqueness
in updating DataSet to database, 3-131
uniqueness in DataRows, 3-131
unmanaged DLLs
mismatch, 2-5
search order, 2-4
unmanged DLLs
config support, 2-4
unsupported SQL commands, 4-6
updating
LOBs, 3-85
updating a DataSet obtained from a REF
CURSOR, 3-74
updating LOBs using a DataSet, 3-85
updating LOBs using ODP.NET LOB objects, 3-86
updating LOBs using OracleCommand and
OracleParameter, 3-85
updating without PrimaryKey and
Constraints, 3-132
user-defined types, 3-99
using FetchSize property, 3-70

V

Value property, 3-50
VARRAY types, 3-100
Virtual Private Database(VPD), 3-17
Visual Studio
documentation, 2-3

W
WCF Data Services, xxiv
web.config, 2-5
Windows registry, 2-12

X
XML
characters with special meaning, 3-90
data manipulation using, 3-95
XML data
saving changes using, 3-96
updating in OracleXmlType, 3-90
XML Database, 3-86
XML DB, 3-86, 3-88
XML element name
case-sensitivity in, 3-97
XML Element Name to Column Name
Mapping, 3-97
XML related classes, 6-1
XML related enumerations, 6-1
XML Support, 3-86
XML to specify custom type mappings, 3-104
XMLQuery, 3-87
XMLTable, 3-87
XMLType column
as a .NET String, 3-89
fetching into the DataSet, 3-89
updating with OracleCommand, 3-89
XMLType columns
setting to NULL, 3-89
XQUERY, 3-87
XQuery
support, 3-88
XQuery language, 3-87